摘 要

可编程控制器作为工业自动化三大支柱产业之一，以其性能优越、可靠性高、易于开发和维护等优点，在工业生产领域发挥着越来越重要的作用。

本文在总结分析可编程控制器的基本构成、工作原理及其特点的基础上，本着顺应可编程控制器的发展趋势以及方便、实用、可靠等设计原则，提出了一种基于 ARM 微控制器的可编程控制器设计方案。论文主要从以下两个部分展开论述：系统硬件设计，系统软件设计（包括 μC/OS-Ⅱ嵌入式操作系统和应用软件）。系统硬件部分先对 ARM 微控制器 LPC2134 作了简要介绍，然后具体介绍了系统电路设计、输入输出电路设计、温度信号采集电路、串行通信接口电路、键盘/显示接口电路、JTAG 接口电路以及 CAN 总线接口电路。软件部分对嵌入式实时操作系统 μC/OS-Ⅱ的移植进行了分析，并给出了移植到本文设计的硬件平台的详细过程；接着详细探讨了系统的软件设计，主要内容有集成开发环境的介绍、启动代码设计以及系统程序设计等。此外，在软件方面还采用了 PID 算法以及软件抗干扰技术。最后，论文对研制工作进行了总结，同时指出了系统设计的不足和改进思路，为课题的进一步研究做了准备。

本课题研制的可编程控制器配置简单，扩展方便，抗干扰能力强，可靠性高。能够采集 4~20mA/0~5V 的模拟量以及 12 路开关量；输出 1 路-10~+10V、4 路 0~5V 与 2 路 0~20mA 的模拟量以及 8 路开关量；能够采集 6 路温度信号；可以应用于开关量的逻辑控制；能实现简单的 PID 控制；并配有 RS232 串行通信接口以及 CAN 总线通信接口。总体上，能满足基本工业控制的要求。

关键词：可编程控制器，ARM 微控制器，LPC2134，μC/OS-II，CAN 总线

Abstract

Programmable Controller, one of the three major pillars in Automation Industry, not only owns excellent performance and high reliability, but is easy to be developed and maintained, plays more and more important role in the filed of industry manufacture.

According to analysis and summarization of the basic structure, principle and features, the design of Programmable Controller based on ARM MCU, is introduced in the thesis, following the tendency of Programmable Controller and the design principle of convenience, good practicability and high reliability. This thesis is consisted of two parts: the design of hardware, the design of software (include μC/OS-Ⅱ embedded operating system and application software). In the hardware part, first ARM MCU LPC2134 is introduced briefly, then the system circuit design, the input and output interface circuit design, temperature signal acquisition circuit , serial communication interface circuit, keyboard/display interface circuit, JTAG interface circuit and CAN bus interface circuit, are introduced respectively. In software part, the transplantation of μC/OS-II embedded real time operation system is analyzed, and the specific process of porting is introduced. Then the software design of the system is discussed in details, including the introduction of the integrated development environment, the design of startup code, the development of system program. Besides, the PID arithmetic and the method of software anti-jamming are applid. Finally, the summary of research wok is supplied; meanwhile, the shortages and the perfectness idea is put forward; and the further research in future is prepared.

The designed Programmable Controller is simple for configuration, conveninent for extention, and owns good performance in anti-jamming and reliablity. 4~20mA/0~5V analog signal and 12-channel switch signal can be collected, and 1-channel -10~+10V/4-channel 0~5V/2-channel 0~20mA analog signal and 8-channel switch signal output can be realized. Besides, 6-channel temperature signal

can be collected. It can be applied to the logical control of the switch signal, and simple PID-control can be realized. Moreover, with RS232 serial communication interface and CAN bus interface. In a word, this Programmable Controller can meet the need of industrial control basically.

Keywords: Programmable Controller，ARM Micro-controller Unit, LPC2134, μC/OS-II，CAN bus

学位论文独创性声明

本人郑重声明：

1、坚持以“求实、创新”的科学精神从事研究工作。

2、本论文是我个人在导师指导下进行的研究工作和取得的研究成果。

3、本论文中除引文外，所有实验、数据和有关材料均是真实的。

4、本论文中除引文和致谢的内容外，不包含其他人或其它机构已经发表或撰写过的研究成果。

5、其他同志对本研究所做的贡献均已在论文中作了声明并表示了谢意。

作者签名：张明华

日　　期：2007.5.20

学位论文使用授权声明

本人完全了解南京信息工程大学有关保留、使用学位论文的规定，学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版；有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆被查阅；有权将学位论文的内容编入有关数据库进行检索；有权将学位论文的标题和摘要汇编出版。保密的学位论文在解密后适用本规定。

作者签名：张明华

日　　期：2007.5.20

第一章 绪 论

1.1 前言

可编程控制器(Programmable Controller, PC)，又称为可编程逻辑控制器(Programmable Logic Controller，PLC）是一种新型的工业控制器，是以微处理器为核心的工业生产自动控制装置。国际电工委员会（IEC）对可编程控制器作了如下定义："可编程控制器是一种数字运算操作的电子系统，专为在工业环境应用而设计的。它采用可编程序的存储器，用来在其内部存储执行逻辑运算、顺序控制、定时、计数与算术运算等操作的指令，并通过数字式、模拟式的输入和输出，控制各种类型的机械或生产过程。可编程控制器及其有关外部设备，都应按易于使工业控制系统形成一个整体，易于扩充其功能的原则设计。"[1]它将传统的继电器控制技术、计算机技术和通信技术融为一体，专门为工业控制而设计，具有功能完善、通用灵活、可靠性高、环境适应性好、编程简单、使用方便等优点，在工业上得到了越来越广泛的应用。

传统的顺序控制系统主要是继电—接触器控制系统，是一种有触点的控制系统，采用的是布线连接方式，其系统不易变更、功能不易扩展，只能用于要求不高的专用逻辑控制场合。当控制对象比较复杂时，其可靠性和难扩充性难以满足实际生产的要求。从而，需要有一种先进的自动控制装置以满足迅速发展的工业生产要求。1968 年，美国通用汽车公司提出了使用新一代控制器的设想。从用户角度考虑，该公司对新一代控制器提了 10 点要求，为各大公司提供了明确的开发目标。1969 年，美国数字设备公司（DEC 公司）研制出了第一台可编程控制器 PDP—14，在美国通用汽车公司的生产线上试用成功，并取得了满意的效果，新一代工业控制设备——可编程控制器自此诞生。[2]

经过近 40 年的发展，PLC 的功能越来越强大。除基本的逻辑控制、定时、计数、算术运算等功能外，还开发了特殊功能模块。常见的特殊模块有高速计数模块、各种定位模块、速度控制模块、各种闭环控制模块。配合这些特殊功能模块，PLC 的应用领域得到不断扩大，广泛地应用于数据处理、位置控制、过程控制、运动控制等领域。目前，PLC、CAD/CAM 和机器人技术已发展成为现代工业自动化的三大支柱。[1]

1.2 PLC 的发展历程

PLC 从 1968 年产生以来，随着微电子技术、计算机技术、自动控制技术、通信技术的发展而迅速发展，其发展过程大致为 ：

第一代 PLC（1969—1972 年)：大多用一位机开发，用磁芯存储器存储；功能简单，主要是逻辑运算、定时、计数；机种单一，没有形成系列。典型产品有：美国 DEC 公司的

PDP-14。

第二代 PLC（1973—1975 年）：元件上采用了 8 位微处理器和半导体存储器（EPROM）；功能上增加了数字运算传送、比较及模拟量控制等；产品初步形成系列。典型产品有：美国 MODICON 公司的 184、284、384，德国 SIEMENS 公司的 SIMATIC S3 系列等。

第三代 PLC（1976—1983 年）：20 世纪 70 年代后期超大规模集成电路和高性能微处理器的出现及引入 PLC，PLC 的功能及处理速度大大增加，增加了浮点数运算、平方、三角函数、相关数、查表、列表、脉宽调制变换等功能，还增加了远程 I/O、一些特殊功能模块和通信、自诊断等功能。典型产品有：德国 SIEMENS 公司的 SIMATIC S5 系列。

第四代 PLC（1984—20 世纪末期）：进入 20 世纪 80 年代中、后期以来，超大规模集成电路迅速发展，微处理器价格大幅度下跌，PLC 中 16 位、32 位片式高性能微处理器全面使用。而且一台 PLC 中配置多个 CPU，进行多通道处理。随着各种智能模块的产生以及编程语言的丰富，PLC 已发展成为一种具有逻辑控制、过程控制、运动控制、数据处理、联网通信等功能的名副其实的“多功能控制器”。[3]典型产品有：德国 SIEMENS 的 SIMATIC S7 系列，美国 A-B 公司的 PLC—5 系列等。

1.3 国内外 PLC 的状况和发展趋势

自从美国研制出世界上第一台 PLC 以后，日本、德国、法国等国相继研制了各自的 PLC。随着 PLC 的不断发展，PLC 已逐步在工业控制领域中占据主导地位。它已广泛用于钢铁冶金、机械加工、汽车制造、石油化工等几乎所有的工业领域。作者通过查阅资料，对国内外 PLC 市场及研究现状进行总结，并将关于 PLC 未来发展趋势的权威观点概括阐述如下。[4]

1.3.1 国外 PLC 的状况

目前，世界上几个工业发达国家，如美国、德国、日本、英国、法国等都有几十家工厂上千种各类 PLC 产品，PLC 在工业企业中应用已相当普及，在多种自动化设备中占首位。美国 Allen-Bradley（A-B）公司，德国西门子（SIEMENS）公司，日本三菱（MITSUBISHI）、欧姆龙（OMRON）等公司都是国际上著名的 PLC 生产厂商，其中 A-B 公司的 PLC—5 系列，功能齐备的各种模块是通用的，处理器模块内集成有通信机制及多种通信接口；SIEMENS 公司的 S7—400 系列大型 PLC，适宜于自动化生产和过程中做高级控制应用。易于扩展，具有强大的通信能力，是中、高档性能控制领域中首选的理想解决方案；[6] 三菱公司推出的 A 系列 PLC 是一种新型的带有智能接口的 PLC，具有控制多模拟量系统的 PID 回路调节功能，并具有很强的通信能力。总之，国外的这些 PLC 产品体现了当今 PLC 技术的最高水平。

1.3.2 国内PLC的状况

国际上PLC的发展经历了从研制、开发、生产到应用4个阶段，而我国PLC的发展是从应用开始的，经历了从成套设备引进应用、PLC产品引进应用、合资生产产品、消化移植产品到PLC产品广泛推广应用几个阶段。[6]我国在PLC研制方面起步较晚，加上整个国家的工业化水平和工艺能力，特别是微电子技术，大规模、超大规模集成电路的设计、生产能力，专用芯片的设计生产能力，与发达的工业国家有明显差距，因此，我国的PLC技术也与先进国家存在一定的差距。目前我国自主研制开发的PLC多为中小型PLC，功能、质量和可靠性等方面也有了明显的提高。具有代表性的产品有南京嘉华JH200系列、北京和利时Hollias-PLC系列等，其中，和利时Hollias-PLC系列PLC的数字量I/O可达到1024点，模拟量I/O可达256点，内置TCP/IP通信接口，配有PROFIBUS-DP现场总线主站及从站和远程I/O。[6]尽管如此，国产PLC的市场占有率仍不超过10%，也没有形成主流产品。

1.3.3 PLC的发展趋势

随着科学技术的进一步发展，现代的PLC也将进一步往前发展，功能越来越多，集成度越来越高，网络功能越来越强，从而也使得PLC的应用领域不断扩展。总体来讲，PLC的发展趋势主要体现在以下几个方面：

1．大型PLC不断向高速度、大容量和多功能方向发展

大型PLC向高速度和多功能方向发展，是使之能取代工业控制微机的部分功能，对大规模、复杂系统进行综合自动控制。存储容量的提高是为大规模系统的设计提供条件，目前大型PLC的存储容量是几百KB，最高可达到几MB。

2．PLC向高性能微小型化方向发展

PLC向微小型方向发展主要表现是向高性能、智能化、模块化、整体型发展。微小型PLC的I/O点数一般在8~128点数字量I/O以下，除了开关量I/O以外，还可以扩展连续模拟量I/O及其他各种特殊功能模块。发展微小型PLC是适应单机控制以及小型自动化的需要，同时，也能更广泛地取代继电器控制。

3．产品更加规范化、标准化

用户促使生产厂家把PLC做成兼容产品，至少PLC的基本部件技术规格、输入输出模块以及通信协议将规范化、标准化，且能相互兼容。在PLC系统结构不断发展的同时，编程软件也在不断发展。PLC最常用的编程语言是梯形图语言。按照IEC61131-3国际标准程序设计语言，包括梯形图、顺序功能图表、功能块图表、结构化文本和指令表等程序设计语言。此外许多公司推出了多种高级语言（如C语言）编程，未来的PLC编程工具和编程语言将规范化、标准化且相互兼容。

4. 加强联网和通信功能

加强联网通信功能包括PLC与计算机之间，不同PLC之间，PLC与现场总线之间通信能力的加强，是现代化工业生产的需要。PLC网络发展趋势符合国际工业标准的开放体系结构，具有高速、层次灵活、高可靠性、大吞吐量等特点，适应多网络兼容连接和最级集成。通信性能方面，主要面向于Ethernet技术和基于Web技术。

5. 新型PLC—软PLC

随着标准IEC61131-3的推广和开放式工业计算机系统的发展，使得PC有可能替代传统的PLC，成为新型的PLC—软PLC。软PLC是基于IPC或EPC的开放结构的控制系统。它提供了与硬PLC同样的功能，利用软件技术将标准的工业PC转换成全功能的PLC过程控制器。软PLC采用开放式结构，将PLC软件开发工具与系统硬件分离，解除了硬件设备对软件的制约，充分利用PC机资源，提供了高速数据处理能力和强大的网络功能。可以满足控制系统的开放性和柔性的要求，将控制、通信功能融为一体，具有广泛的发展前景。

1.4 本课题研究的主要内容

本论文的任务是根据可编程控制器的系统组成与工作原理，利用嵌入式微控制器高性能、低功耗、低成本的特点，设计一个以ARM微控制器LPC2134为核心的可编程控制器。该可编程控制器具有开关量I/O控制、模拟量I/O控制以及PID控制等功能。此外，为了适应PLC网络发展的方向，本设计增加了CAN总线接口电路。整个设计以LPC2134为核心，配置相应的外设及接口电路，用C语言以及汇编语言开发，组成一个配置简单、扩展方便、功耗低、可靠性高的微小型可编程控制器。

基于ARM的可编程控制器的研制主要涉及硬件设计和软件设计。

1. 硬件设计。 完成系统总体硬件设计，并对具体实现电路进行详细的分析和设计。硬件电路包括开关量I/O电路、模拟量I/O电路、温度信号采集电路、显示通讯电路以及电源电路等。

2. 软件设计。 系统的具体实现，对系统按功能模块进行介绍。主要包括 μC/OS-II操作系统的移植，监控程序的模块化设计，PID控制算法的设计，并结合SmartARM2200开发实验板，利用ADS1.2集成开发工具对ARM嵌入式系统的软硬件进行仿真调试。

第二章 可编程控制器的系统分析

本章首先介绍 PLC 的基本组成及各组成部分的作用，然后进一步分析 PLC 工作原理。

2.1 PLC 的基本组成及各组成部分的作用

2.1.1 PLC 的基本组成

PLC 是一种以微处理器为核心的用于工业自动控制的工业控制器，其本质上是一台工业控制专用计算机，所以它的组成与一般的微型计算机相类似。PLC 硬件主要由中央处理单元（CPU）、存储器、输入/输出接口、电源、编程器以及智能输入/输出接口等构成。PLC 硬件结构框图如图 2.1 所示。

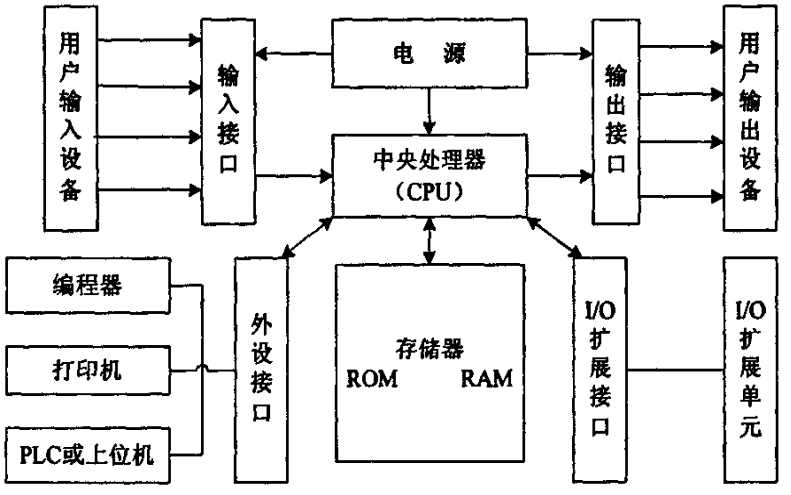

图 2.1　PLC 硬件结构框图

2.1.2 PLC 各组成部分的作用

1. 中央处理单元（CPU）

与一般计算机一样，CPU 是 PLC 的核心，它的主要功能有：按照系统程序所赋予的功能，接收并存储从编程器输入的用户程序和数据；用扫描的方式通过 I/O 部件接收现场的状态或数据，并存入输入映像存储器或数据存储器中；诊断 PLC 内部电路的工作故障和编程中的语法错误等；执行各种运算程序；输出运算结果；与外部设备或计算机通信等。

2. 存储器

PLC 存储系统包括系统存储器和用户存储器两部分。系统存储器用于存储 PLC 内部信

息，包括程序存储器ROM或EPROM和数据存储器RAM或EEPROM，其中ROM或EPROM用于存储系统监控程序，而RAM用做内部继电器（软继电器）、移位寄存器、数据寄存器、定时/计数器以及累加器等；[7]用户存储器用来存放用户针对具体控制任务用规定的PLC编程语言编写的各种用户程序。

3. 输入/输出接口

输入/输出接口是PLC与外界连接的接口。PLC通过各种I/O接口接收所需被控对象的各种信息，又将处理结果输送给外部被控对象，驱动各种执行机构，实现对被控对象的控制。输入接口用来接收和采集开关量输入信号和模拟量输入信号，输出接口用来连接被控对象中各种执行元件。

4. 电源单元

电源单元是PLC的电源供给部分，它将交流电源转换成系统内部各单元所需的直流电源，使PLC能正常工作。电源内部电路由多级滤波、稳压电路等构成，能克服电网波动、温度漂移等因素的影响，并对电路具有一定的保护能力，防止电压突变时损坏中央控制单元。

5. 通信接口

PLC配有多种标准通信接口或网络接口，以实现PLC与PLC之间的链接或互连，或者实现PLC与其他具有标准通信接口的设备之间的连接，通过这些通信接口可以与编程器、人机界面、打印机及计算机等相连。与其他PLC相连时，可以组成多机系统或网络，实现更大规模的控制。当与计算机相连时，可以组成多级分布式控制系统，实现控制与管理相结合的综合系统。通信接口一般是RS232或RS422或RS485串行通信接口以及可以选配的现场总线通信模块。

6. PLC的外部设备

PLC的外部设备包括编程器、打印机、EPROM写入器等。其中编程器用来输入、编辑、调试用户程序和监视PLC的运行。现在PC已普遍用作PLC的通用编程器使用，通过RS-232通信口与PLC相连，在PC上进行梯形图编辑、调试和监控，可实现人机对话、通信及打印，使编程及调试更为快捷便利。

7. 智能输入/输出接口

根据PLC应用的各种特殊功能的需要，PLC配有多种智能I/O接口。例如高速计数模块、PID控制模块、温度传感器模块等。这些智能I/O单元是一个独立的自治系统，通过系统总线与主机相连，可在主机CPU的协调管理之下独立工作。

2.2 PLC 的工作原理

2.2.1 PLC 的工作方式

PLC 是一种工业控制计算机，它的原理是建立在计算机原理基础上的，即通过执行反映控制要求的用户程序来实现的。为了便于执行程序，在存储器中设置输入映像寄存器区和输出映像寄存器区，分别存放执行程序之前的各输入状态和执行过程中各结果的状态。PLC 的工作方式是一个不断循环的顺序扫描工作方式。CPU 从第一条指令开始，在无中断或跳转控制的情况下，按顺序逐条扫描用户程序，直到程序结束，即完成一个扫描周期，然后返回第一条指令开始新的一轮扫描。PLC 就这样周而复始地重复上述循环扫描。由于 CPU 的运算处理速度很高，使得从外观上看，用户程序似乎是同时执行的。

2.2.2 PLC 的系统工作过程

PLC 在运行时，内部要进行一系列操作，大致包括以下几个方面内容：初始化处理；系统的自诊断；通信服务；现场 I/O 数据处理以及执行用户程序。PLC 的整个工作过程可用图 2.2 表示。

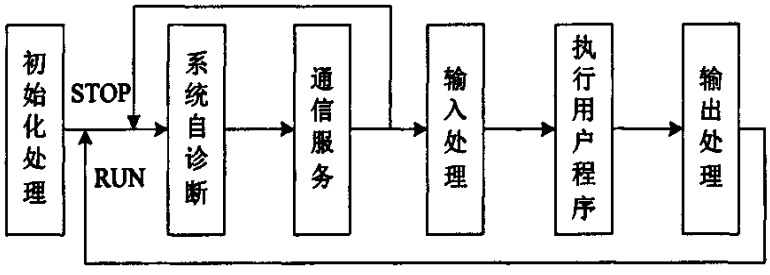

图 2.2 PLC 工作过程示意图

1. 初始化处理：PLC 上电后，首先进行系统初始化，进行清零或复位处理，以消除各元件状态的随机性，同时对电源、PLC 内部电路、用户程序的语法进行检查。

2. 系统自诊断：PLC 在每个扫描周期都要进入系统自诊断阶段，以确保系统可靠性。自诊断程序定期检查 CPU、程序存储器、I/O 单元、通信等是否正常，定期复位监控定时器等。

3. 通信服务：通信服务指的是与编程器、PLC 的其他外设、网络（配置有网络通信时）等进行信息交换。在这个阶段，进行 PLC 之间以及 PLC 与计算机之间的信息交换；PLC 与其他带微处理器的智能装置通信等。[8]

4. 执行用户程序：PLC 在运行状态下，每一个扫描周期都要执行用户程序。PLC 的用

户程序由若干条指令组成，指令在存储器中按顺序排列。在用户程序执行阶段，在没有跳转指令时，CPU从第一条指令开始，逐条顺序地执行用户程序，并把运算结果存入输出映像区对应位中。

5. 输入、输出处理：PLC在运行状态下，每一个扫描周期都要进行输入、输出处理。输入处理就是对PLC的输入进行一次读取，将输入端各变量的状态重新读入PLC中，存入输入映像区；输出处理就是将运算后的结果存入输出映像区，直至传送到外部被控设备。

整个工作过程中，当PLC处于STOP（停止）状态时，只完成系统自诊断和通信服务工作；当PLC处于RUN（运行）状态时，除了完成系统自诊断和通信服务工作外，还要完成输入处理、用户程序执行、输出处理工作。

2.2.3 PLC程序循环扫描过程

PLC程序循环扫描过程主要分为三个阶段，即输入采样阶段、程序执行阶段与输出刷新阶段，下面对这三个阶段的情况进行较详细的分析。

1. 输入采样阶段：PLC在输入采样阶段扫描所有输入端子，将各输入状态存入内存中各对应的输入映像寄存器中。此时，输入映像寄存器被刷新。在程序执行阶段和输出刷新阶段，无论输入信号状态如何变化，输入映像寄存器的内容都不会变化，直到下一个扫描周期的输入采样阶段，输入映像区中的内容才会被刷新。

2. 程序执行阶段：PLC按先左后右、先上后下的次序，逐条执行程序指令，从输入映像寄存器中和其他元件寄存器读出有关状态。然后，进行相应的运算，运算结果再存入元件映像寄存器中。此外，各元件映像寄存器中的内容，随着程序的执行在不断变化。

3. 输出刷新阶段：在所有指令执行完毕后，CPU将元件映像寄存器中所有输出继电器的状态转存到输出锁存器中，通过一定方式输出，驱动外部负载。从而完成本周期运行结果的实际输出。

第三章 嵌入式系统分析与设计

3.1 嵌入式系统简介

嵌入式系统（Embedded System）是嵌入式计算机系统的简称，它是嵌入到目标体系中的专用计算机系统。具体地讲，嵌入式系统是指以应用为中心，以计算机技术为基础，并且软硬件可裁剪，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统。[9]从嵌入式系统的组成来看，一个嵌入式系统由硬件、软件、开发工具和开发系统 4 部分组成。其中嵌入式硬件包括嵌入式处理器/控制器/数字信号处理器（Digital Signal Processor，DSP）等、存储器及外设器件、输入输出（I/O）端口、图形控制器等；嵌入式软件部分主要有嵌入式操作系统和应用软件。嵌入式系统的硬件和软件位于嵌入式系统产品本身，开发工具则独立于嵌入式系统产品之外。开发工具一般用于开发主机，包括语言编译器、连接定位器、调试器等，它们构成了嵌入式系统的开发系统与开发工具。

目前，嵌入式系统已广泛地应用于智能产品、工业控制、办公自动化、石油化工、网络通信、汽车电子、航空航天以及军事等各个领域。

3.2 嵌入式处理器

3.2.1 嵌入式处理器概述

嵌入式系统的核心部件是嵌入式处理器。据不完全统计，到目前为止，全世界嵌入式处理器的品种总量已经有 1000 多种，流行体系结构有 30 多个系列，其中 8051 体系占了多半。

嵌入式处理器一般具备以下 4 个特点：（1）对实时性和多任务有很强的支持能力，能完成多任务并且有较短的中断响应时间；（2）具有功能很强的存储区保护功能；（3）可扩展的处理器结构；（4）嵌入式微处理器的功耗很低。[10]

目前，嵌入式处理器主要有嵌入式微处理器（EMPU）、嵌入式微控制器（EMCU）、嵌入式数字信号处理器（EDSP）以及片上系统（SoC）四类。其中嵌入式微控制器一般以某种微处理器内核为核心，芯片内部集成 ROM、RAM、定时器、I/O、串行口、A/D 等必要的功能模块。微控制器是目前嵌入式系统应用的主流。

3.2.2 ARM 微处理器

ARM 是 Advanced RISC Machines 的缩写，ARM 既可以认为是一个公司的名字，

也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。目前，基于 ARM 技术的微处理器约占了 32 位 RISC 微处理器 75%以上的市场份额。[11]ARM 微处理器已广泛应用于工业控制、消费类电子产品、通信系统、网络系统、无线通讯等各个领域。

目前，ARM 微处理器有 ARM7 系列、ARM9 系列、ARM9E 系列、ARM10E 系列、SecurCore 系列以及 Intel 的 Xscale 等几个系列。其中，ARM7 系列微处理器是低功耗的 32 位 RISC 处理器，该系列具有如下特点：（1）具有嵌入式 ICE 逻辑，调试开发方便；（2）极低的功耗，适合对功耗要求较高的应用；（3）能够提供 0.9MIPS/MHZ 的三级流水线结构；（4）代码密度高，并兼容 16 位的 Thumb 指令集；（5）对操作系统的广泛支持；（6）指令系统与 ARM9 系列、ARM9E 系列和 ARM10E 系列兼容，便于用户的产品升级换代；（7）主频最高达到 130MIPS，高速的运算处理能力能胜任绝大多数的复杂应用。目前，ARM7 微处理器的主要应用领域为：工业控制、Internet 设备、网络和调制解调器设备、移动电话等多媒体和嵌入式应用。[12]

ARM7 系列微处理器包括如下几种类型的核：ARM7TDMI、ARM7TDMI-S、ARM720T、ARM7EJ。其中，ARM7TDMI 是目前使用最广泛的 32 位嵌入式 RISC 处理器内核，属低端 ARM 处理器核。ARM7TDMI 的后缀意义如下：T：支持 16 位压缩指令集 Thumb；D：支持片上 Debug；M：内嵌硬件乘法器（Multiplier）；I：嵌入式 ICE，支持片上断点和调试点。

3.3 嵌入式操作系统

嵌入式操作系统 EOS（Embedded Operating System）是嵌入式应用软件的开发平台，负责嵌入式系统的全部软、硬件资源的分配、调度、控制、协调；它必须体现其所在系统的特征，能够通过加载/卸载某些模块来达到系统所要求的功能。嵌入式操作系统是嵌入式系统的灵魂，它使得嵌入式系统的开发效率大大提高，系统开发的总工作量大大减少，同时也提高了嵌入式软件的可移植性。为了满足嵌入式系统的要求，嵌入式操作系统必须包含操作系统的一些最基本的功能，用户可以通过 API 函数来使用操作系统。

常见的嵌入式操作系统有：嵌入式 Linux、Windows CE、VxWorks、OSE、Nucleus、eCos、μC/OS-II。其中，μC/OS-II 是一个源码公开、可移植、可固化、可裁剪、占先式的实时多任务操作系统，主要用于中小型嵌入式系统。该操作系统支持多达 64 个任务，大部分嵌入式微处理器均支持 μC/OS-II。

3.4 嵌入式系统设计

3.4.1 嵌入式系统的设计步骤

嵌入式系统设计一般由需求分析，体系结构设计，硬件/软件/执行机构设计，系统集成

及系统测试 5 个阶段组成。各个阶段之间要求不断反复和修改，直至完成最终设计目标。[9]

其中，嵌入式系统的系统需求分析就是确定设计任务和实际目标，并提炼出设计规格说明书，作为正式设计指导和验收的标准；体系结构设计描述了系统如何实现所述的功能和非功能需求，包括对硬件、软件和执行结构的功能划分，以及系统的软件、硬件和操作系统的选型等；硬件设计主要包括电路原理图设计、PCB 布线及元器件选型。在整个嵌入式系统设计过程中，大部分工作都集中在软件设计上，面向对象技术、软件组件技术与模块化设计是现代软件工程经常使用的方法。执行机构的设计主要任务是选型，选择合适的执行机构，配置相应的驱动器以及相关信号调理电路等，并考虑与嵌入式系统硬件的连接方法；系统集成就是把系统的硬、软件和执行装置集成在一起进行调试，发现并修改单元设计过程中的错误，弥补前期设计的不足之处；系统测试的任务就是对设计好的系统进行全面测试，看其是否满足规格说明书中给定的功能要求。

3.4.2 嵌入式系统的设计方法

传统的嵌入式系统开发采用的是硬件开发与软件开发分离的方式，能改善硬、软件各自的性能，但不一定能使系统综合性能达到最佳。虽然在系统设计的初始阶段考虑了软、硬件的接口问题，但由于软、硬件分别开发，各自部分的修改和缺陷容易导致系统集成出现问题。此外，这些错误不但难于定位，而且对于它们的修改会涉及整个软件结构或硬件配置的改动。基于以上问题，一种新的开发方法应运而生，即硬、软件协同设计方法。[9] 硬、软件协同设计方法过程如图 3.1 所示。

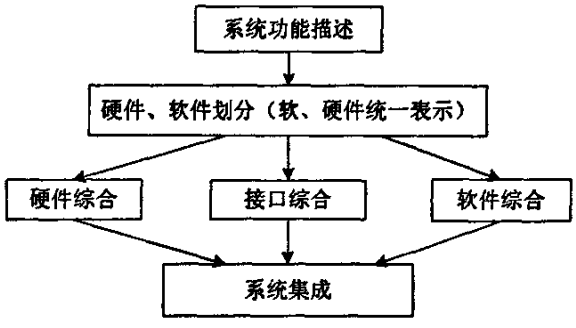

图 3.1 嵌入式系统的硬、软件协同设计方法

硬、软件协同设计过程可归纳为：（1）需求分析；（2）硬、软件协同设计；（3）硬、软件实现；（4）硬、软件协同测试和验证。

3.5 基于 ARM 的 PLC 系统设计方案

结合 PLC 的基本组成与功能要求，作者提出了一种以 ARM 微控制器为核心的嵌入式系统设计方案。该嵌入式系统由硬件和软件两大部分组成，即以 ARM 微控制器 LPC2134 为中心的硬件设计和基于嵌入式实时操作系统 μC/OS-II 的软件开发。

3.5.1 硬件设计的系统分析

在设计过程中采用 ARM 微控制器来研制 PLC，硬件资源配置以 PHILIPS LPC2134 微控制器为核心。该微控制器内部集成了足够的 RAM，FLASH ROM，UART 接口和 I^2C 总线接口等。于是该 PLC 硬件系统可以简化成：CPU+接口，能够大大降低成本，提高硬件工作可靠性，同时由于 LPC2134 配置了标准的 JTAG 接口，支持系统对 CPU 进行在系统编程和调试，这对将来 PLC 产品的在系统编程、调试功能提供了有力保障。外围电路配置有开关量输入输出电路，模拟量输入输出电路，温度信号采集电路，电源电路，串口通信接口电路，键盘显示电路等。此外，为了适应 PLC 的网络发展方向，增加了 CAN 总线接口电路，采用独立的 CAN 控制器 SJA1000、 CAN 总线收发器 PCA82C250 以及高速光电耦合器 6N137。

3.5.2 软件设计的系统分析

在嵌入式系统软件开发过程中，最重要的环节就是操作系统的选择。只有确定了嵌入式操作系统，才可以根据系统的功能目标，设计出相应的应用软件系统和软件模块。

基于 ARM 的可编程控制器的系统软件采用了嵌入式实时操作系统 μC/OS-II，因此，整个系统的应用程序都是基于 μC/OS-II 操作系统的。为了使 μC/OS-II 在基于 LPC2134 的嵌入式系统上能够正常运行，需要完成 μC/OS-II 操作系统的移植。然后，利用 μC/OS-II 操作系统提供的 API 函数以及 ARM 核微控制器集成开发工具 ADS1.2 开发 PLC 系统的监控程序。其中，PLC 监控程序按照模块化方法可分为键盘处理程序、显示程序、A/D 与 D/A 转换程序以及通信程序等；此外，还涉及到 PID 控制算法的实现。

第四章　基于 ARM 的可编程控制器的硬件设计

根据上一章提出的 PLC 系统设计方案，下面就对本课题已经完成的系统硬件设计过程进行说明，并对具体的硬件设计电路展开分析。

4.1 硬件设计系统框图

基于 ARM 的可编程控制器的硬件设计主要包括 ARM 微控制器 LPC2134、开关量输入输出单元电路、模拟量输入输出单元电路、温度信号采集电路、RS232 通信接口电路、电源电路、时钟与复位电路、键盘显示电路、JTAG 接口电路、CAN 通信接口电路等。硬件设计系统框图如图 4.1 所示。

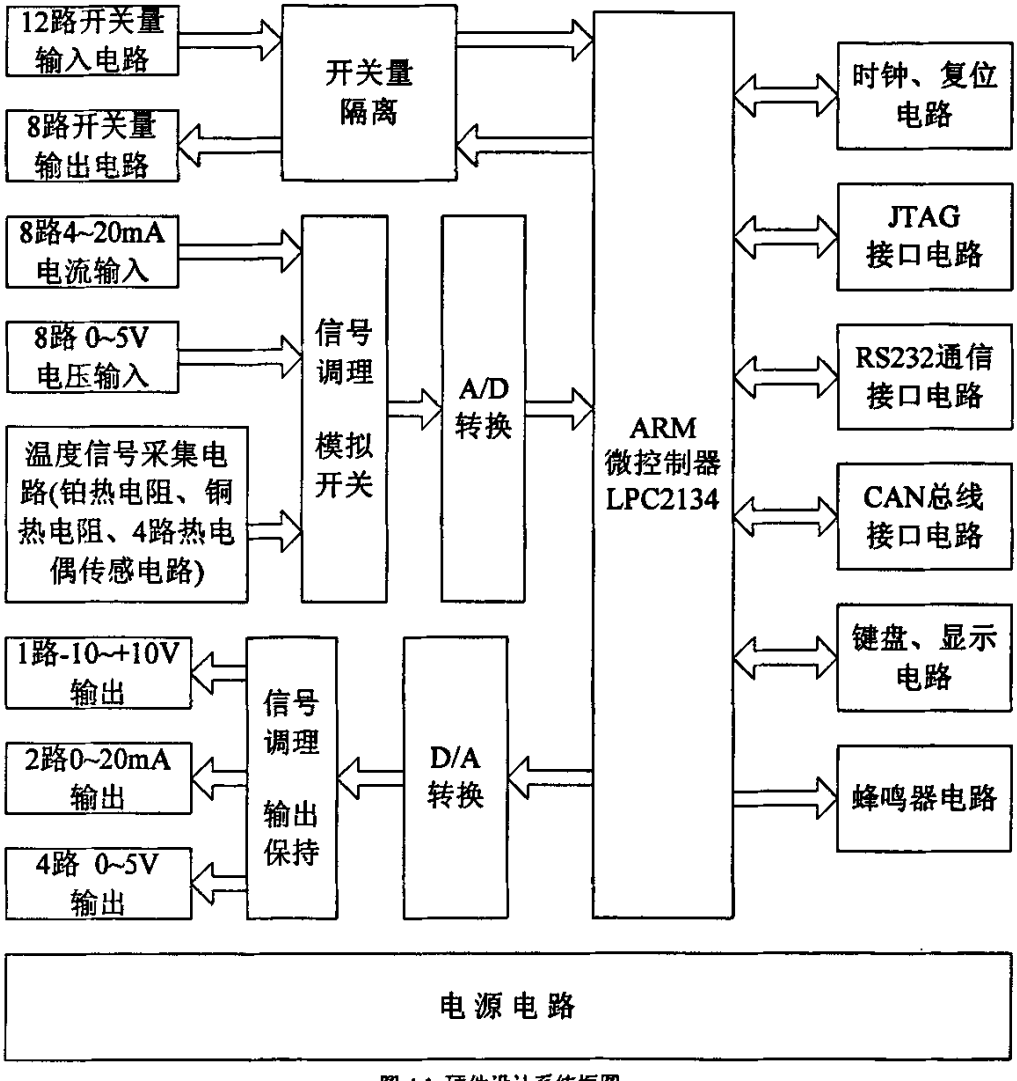

图 4.1 硬件设计系统框图

4.2 LPC2134 芯片简介

LPC2134 是基于一个支持实时仿真和跟踪的 16/32 位 ARM7TDMI-S CPU 的微控制器。它具有如下主要特性：[13]

小型 LQFP64 封装的 16/32 位 ARM7TDMI-S 微控制器

16KB 片内静态 RAM，128KB 片内 Flash 程序存储器；128 位宽度接口/加速器实现高达 60MHz 的操作频率。

片内 Boot 装载程序实现在系统编程（ISP）和在应用编程（IAP）。

EmbeddedICE-RT 和嵌入式跟踪接口，可实现实时调试和高速跟踪执行代码。

2 个 32 位定时器/计数器和内部看门狗。

多个串行接口，包括 2 个 16C550 工业标准 UART、2 个高速 I^2C 接口、SPI 和具有缓冲作用和数据长度可变功能的 SSP。

多达 47 个可承受 5V 电压的通用 I/O 口；多达 9 个边沿或电平触发的外部中断引脚。

通过片内 PLL 可实现最大为 60MHz 的 CPU 操作频率；片内集成振荡器与外部晶体的操作频率范围为 1~30 MHz，与外部振荡器的操作频率范围高达 50MHz。

2 个低功耗模式：空闲和掉电。

单电源供电，含有上电复位和掉电检测电路。CPU 操作电压为 3.0~3.6 V。I/O 口可承受 5V 的电压。

较小的封装和极低的功耗使 LPC2134 非常适用于小型系统应用中。4 个 32 位定时器、6 个 PWM 通道和多达 47 个 GPIO 以及多达 9 个边沿或电平触发的外部中断，使它们特别适用于工业控制应用和医疗系统。[14]

4.3 系统电路设计

4.3.1 电源电路设计

本系统是多电源系统，涉及到五种共地电源：+5V，-5V，+12V，-12V，以及+3.3V。其中模拟开关 CD4051、A/D 转换器 ICL7135、运算放大器 OP07 等需要+5V，-5V 供电电压；运算放大器 OP07 也需要+12V，-12V 供电电压；此外 D/A 转换器 DAC7512 等需要+12V 供电电压；LPC2134 微控制器的内核以及 I/O 口、串口通信芯片 MAX3232、ZLG7290 等需要+3.3V 供电电压。图 4.2 即为电源电路设计图。

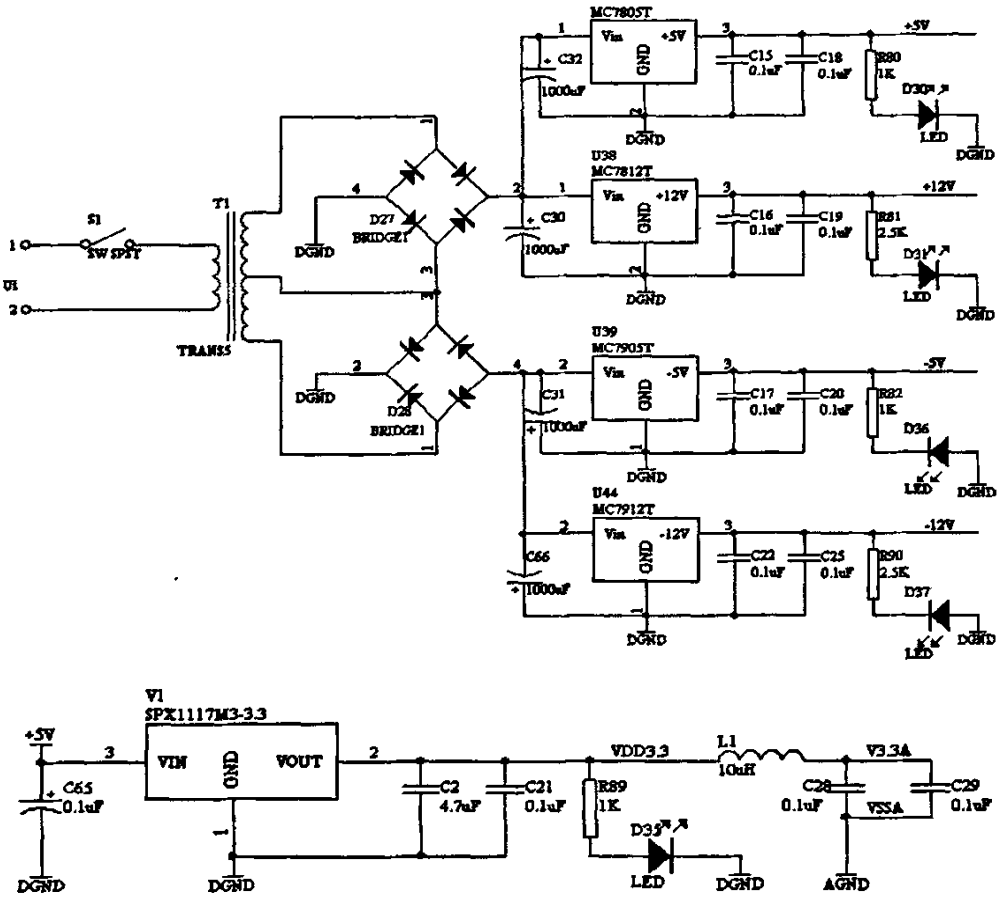

图 4.2 电源电路设计图

本系统中所需电源电压为直流电压，最高值为 12V，稳压块压降为 2V，所以稳压前直流电压为±14V。电网提供的交流电有效值为 220V，变压后交流有效值为 10V，能满足整流为 14V 直流电压。本设计选用三抽头变压器，抽头在次级线圈的中心位置，初级为 220V 次级为±10V。当给电源系统提供 220V 的交流电源时，变压器就输出两组大小相同、相位相反的交流电压，峰值分别为+14V 和–14V。然后通过整流桥，将正负交替的正弦交流电压变换成单方向的脉动电压。根据电源电流与整流桥的要求，选定的整流桥规格为 2A/50V。随后通过滤波稳压产生达到所需要的直流电源。根据电压需求，稳压芯片选用 MC7805[15]、MC7812[15]、MC7905[16]、MC7912[16]和 SPX1117[17]，分别实现+5V，+12V，-5V，+12V，+3.3V 的电压输出。其中 MC7805、MC7812、MC7905、MC7912 集成块输出电压为固定值，即集成块名的后两位，78 系列输出为正电压，79 系列输出为负电压。SPX1117 选用输出电压为 3.3V 的芯片 SPX1117M3-3.3。电路中，三端稳压芯片输入端接有 1000μF 的电容，是为抵消输入时的电感效应，以防止稳压块产生自激振荡，保证正常工作。三端稳压芯片输出端接有 0.1μF 电容，这是为了消除电路的高频噪声，改善负载瞬态的响应。此外，同时为了检测各个电源的输出情况，分别通过不同阻值的电阻串联不同颜色的 LED 来指示不同

电源的工作状态。

需要注意的是，LPC2134 具有独立的模拟电源引脚 V_{DDA}，V_{SSA}，为了降低噪声和出错几率，所以，在 3.3V 电源电路中，用电感 L1 将模拟电源与数字电源进行隔离。

4.3.2 复位监控电路设计

在系统中，复位电路主要完成系统的上电复位和系统在运行时用户的按键复位功能。为了提高系统的稳定性和可靠性，采用看门狗电路监视系统 MPU 的运行，在系统程序出错或因外界干扰而不稳定时及时复位系统，使系统重新运行。

看门狗复位电路可保证由 ARM 微控制器组成的 PLC 能从故障中迅速恢复到正常工作状态。看门狗监控芯片采用 CAT1025，CAT1025 利用低功耗 CMOS 技术将 2K 位的串行 EEPOM 存储器和用于掉电保护的系统电源监控电路集成在一块芯片内。存储器采用 400KHz 的 I^2C 总线接口。

CAT1025 包含 1.6 秒的看门狗定时器电路，可在系统由于软件或硬件干扰而被终止或“挂起”时将系统复位到一个可知的状态。看门狗定时器监控着 SDA 的状态。片内 2K 位的串行 EEPROM 构成 16 字节的页。另外，V_{CC} 电压监控电路提供了硬件数据保护功能，防止在 V_{CC} 降到低于复位门槛电压或上电时 V_{CC} 上升到复位门槛电压之前对存储器的写操作。具体电路如图 4.3 所示。

4.3.3 系统时钟电路设计

目前所有的微控制器均为时序电路，需要一个时钟信号才能工作，因此需要设计时钟电路。LPC2134 微控制器可使用外部晶振或外部时钟源，内部 PLL 电路可调整系统时钟，使系统运行的速度更快（CPU 最大操作时钟为 60MHz）。倘若不使用片内 PLL 功能及 ISP 下载功能，则外部晶振频率范围是 1~30 MHz，外部时钟频率范围是 1~50 MHz ；若使用片内 PLL 功能或 ISP 下载功能，则外部晶振频率范围是 10~25 MHz，外部时钟频率范围是 10~25 MHz 。

在本设计中，LPC2134 使用了外部 11.0592MHz 晶振，电容 C10 与 C11 取 20pF，用 1MΩ 电阻 R138 并接到晶振的两端，使系统更容易起振，并使波特率更精确。同时能支持 LPC2134 微控制器片内 PLL 功能及 ISP 功能。时钟电路如图 4.4 所示。

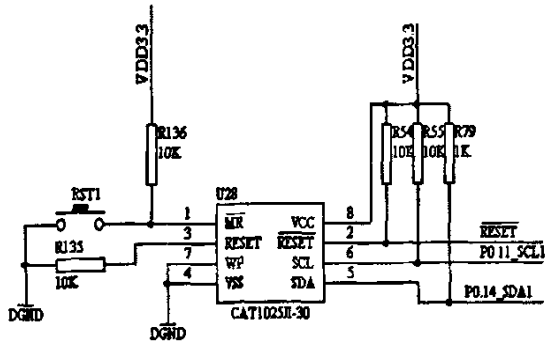

图 4.3 复位监控电路

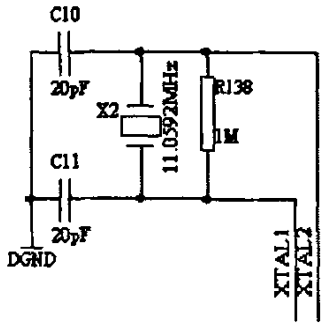

图 4.4 系统时钟电路

4.4 开关量输入/输出电路设计

4.4.1 开关量输入电路设计

开关量输入电路的功能是采集现场各种开关接点的状态信号，并将其转换成标准逻辑电平，送给 CPU 处理。开关量输入电路有直流开关量输入电路和交流开关量输入电路。本设计中，开关量输入电路有 8 路直流开关量输入与 4 路交流开关量输入，下面就分别展开介绍。

1. 直流开关量输入电路

以 4 路直流开关量输入电路为例，对电路设计原理进行分析，电路如图 4.5 所示。其中 COM 为公共端，发光二极管 LED1~LED4 为各输入点的状态指示灯，TLP521-4 为 4 路光电耦合器，它实现现场与 PLC 的 CPU 电气隔离，提高抗干扰性。同时将现场开关量信号转换成符合 CPU 要求的标准逻辑电平。施密特触发器 74LS14 用于消除现场开关按键闭合和断开瞬间的电平抖动。以图 4.5 中第 1 路为例可知，电阻 R1、R2、R19 以及发光二极管 LED1 和 TLP521-4 中的发光二极管组成分压网络，使加在两只发光二极管上的电压在现场开关闭合时为 2V，在现场开关断开时为 0V。这样，电阻 R1、R2、R19、LED1 与 D1 组成的电路就将把外部电路开关的状态变为光耦的发光二极管的发光状态。电阻 R33 与光耦的光敏二极管串联分压。当光耦中发光二极管发光时，分压点为低电平；不发光时，分压点为高电平。

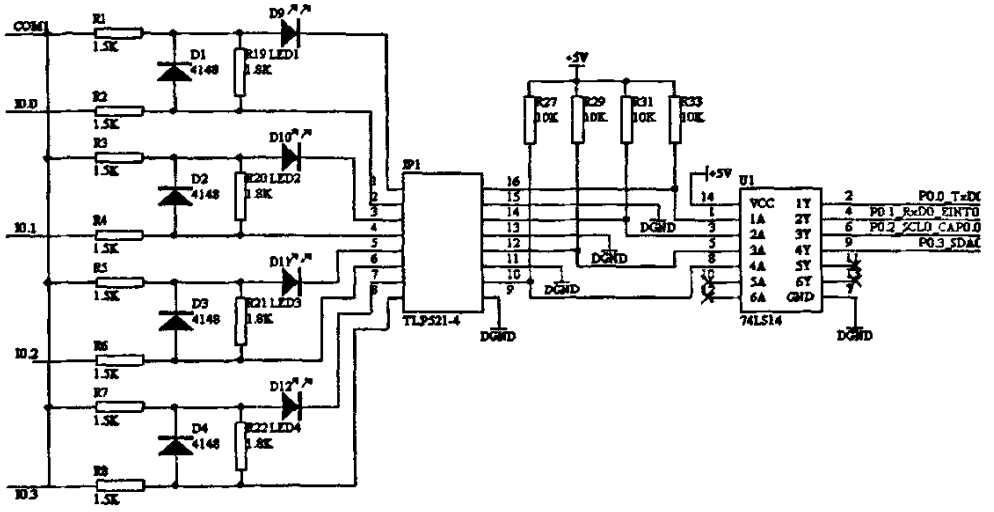

图 4.5　4 路直流开关量输入电路

2. 交流开关量输入电路

交流开关量输入电路将交流信号转换成 CPU 要求的标准逻辑电平。图 4.6 为 4 路 220V 交流开关量输入电路。由图可知，除了信号输入处理部分与直流开关量输入电路不同外，其他部分都相同。该电路有输入滤波电路、限流电路、整流滤波电路、光电隔离电路以及施密特触发器组成。现场提供的 220V 交流电压信号经整流后得到直流电压信号，再经过光电耦合器 TLP521-4 隔离并转换成标准逻辑电平，最后由经施密特触发器 74LS14 供 CPU 读取。

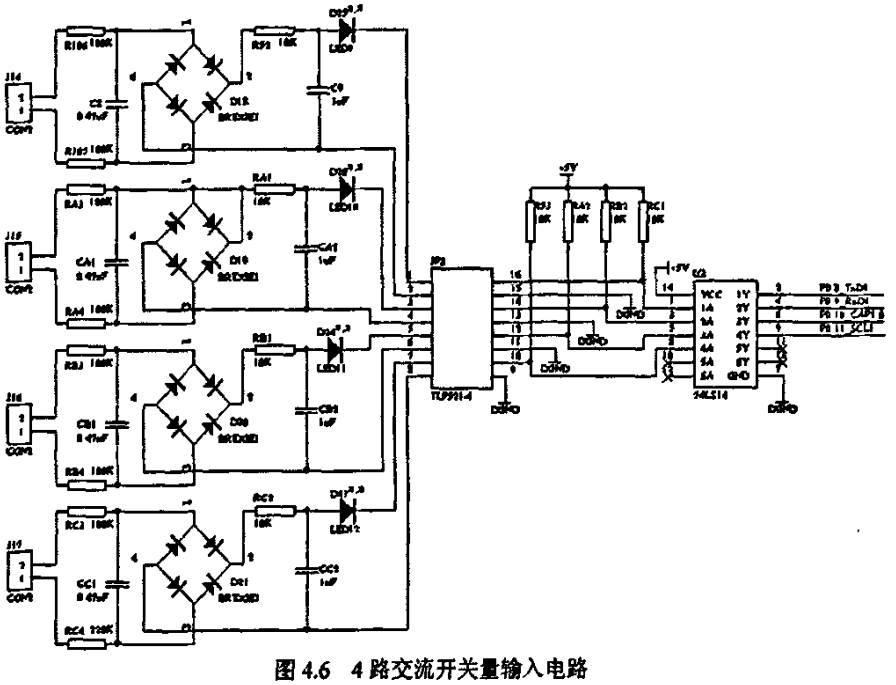

图 4.6　4 路交流开关量输入电路

同样，以图 4.6 中第 1 路为例，R106 与 C8 组成输入信号的交流滤波电路，以提高电路的抗干扰能力；R107 为限流电阻，以限制进入该输入电路的电流；220V 交流信号经过整流桥 D8 得到直流信号；R52 与 C9 组成高频滤波电路对整流后的直流信号进行高频滤波；LED9 用于对外指示该路是否有信号接通；光电耦合器 TLP521-4 将滤波后的直流信号转换成标准逻辑电平，再由经施密特触发器 74LS14 供 CPU 读取。其中，光电耦合器 TLP521-4 还起到电气隔离的作用。

4.4.2 开关量输出电路设计

开关量输出电路是 PLC 与外部连接的输出通道。PLC 通过它向外部现场执行部件输出相应的控制信号。开关量输出电路通常有晶体管输出电路、晶闸管输出电路和继电器输出电路。本设计中，开关量输出电路采用了 4 路晶体管输出与 4 路继电器输出。下面就具体电路进行分析。

1. 晶体管输出电路

晶体管输出电路用于直流负载。以其中 1 路晶体管输出电路为例，对其原理进行简要介绍。具体电路如图 4.7 所示。该电路的核心部件就是作开关用的三极管 9013，其主要作用是作电流放大和电平转换。发光二极管 D32 为输出点的指示灯，二极管 IN4148 是防止端子上+24V 电压极性接反，同时也防止+24V 误接到高电压上或交流电源上而损坏。熔断器 F1 在输出短路或过流时熔断，以保护三极管不被损坏。当 P0.27 为高电平时，光电耦合器驱动三极管 9013，使其饱和导通，负载所需要的大电流由三极管的集电极提供；当 P0.27 为低电平时，光电耦合器的副边侧不输出电流，三极管因没有基极电流而自由截止，此时负载上既无电压，也无电流，即端子上的输出为 0。

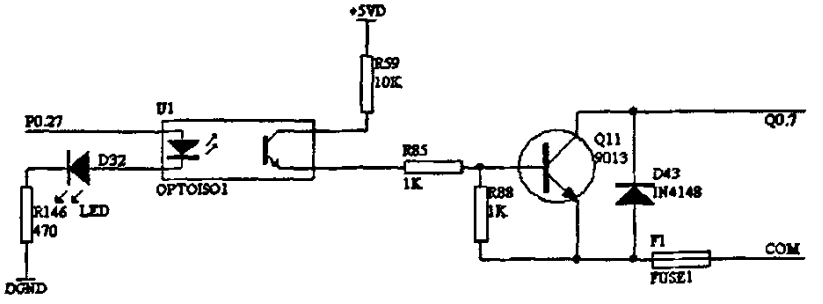

图 4.7 晶体管输出电路

2. 继电器输出电路

继电器输出电路可用于直流负载，也可用于交流负载。它特别适合于对动作时间和工作频率要求不高的场合。继电器是一种通过电磁力控制开关闭合的电器元件，只要在继电器控制线圈端输入高低电平，便可控制开关的闭合。此外，线圈与开关是完全隔开的，因此可在开关端接高低电平，实现利用弱电控制强电的目的。图 4.8 是 4 路继电器输出电路，

由于各路的电路结构完全相同，因此以第 1 路为例加以说明。

其中，ULN2003[18]是达林顿晶体管集成电路，它用作继电器驱动器，可以实现 500mA 集电极电流输出以及高电压输出（50V），并且带有钳位二极管。K1 为继电器，R114 和 C33 组成滤波电路，CON2 为对外输出端。当 P1.28 输入高电平，ULN2003 的 OUT7 输出为低电平，则继电器 K1 得电，其常开触点闭合，外部控制形成回路，负载得电。此继电器输出电路一般输出功率可带 2A 的负载。此外，继电器输出电路响应时间比较慢，从输出继电器的线圈得电（或断电）到输出触点 ON（或 OFF）的响应时间均为 10ms。

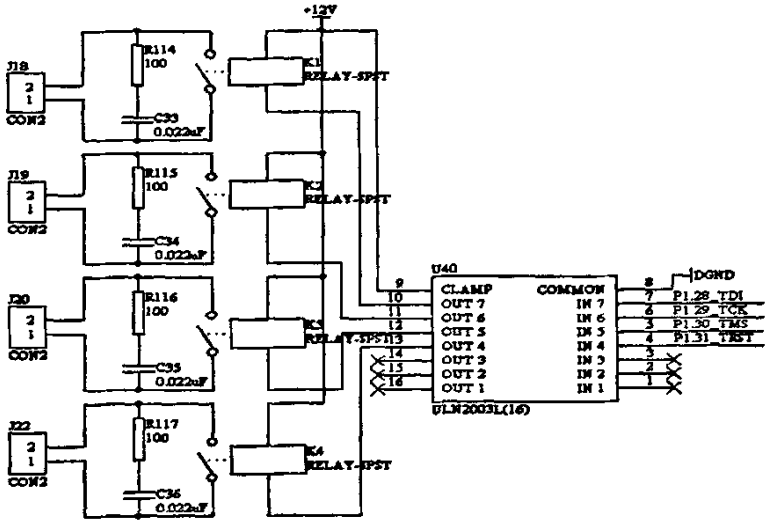

图 4.8　4 路继电器输出电路

4.5 模拟量输入/输出电路设计

4.5.1 模拟量输入电路设计

模拟量输入电路的主要功能就是把现场被测的模拟量信号转变成PLC可以处理的数字量信号。A/D 转换器是模拟量输入电路的主要器件，配合多路模拟开关电路、信号放大调理电路以及相应的控制电路，可以完成模拟量的采样和转换。工作时，PLC 通过总线电路和控制电路发出采样控制信号，通过切换多路开关接通相应的采样通道，使被采样信号通过多路转换开关进入信号放大调理电路，同时 CPU 启动 A/D 转换器对其进行 A/D 转换，转换后的数字量由 CPU 读入 PLC 系统的输入映像缓冲区。从而完成对模拟量的采集。该设计中，模拟量输入电路有 8 路 4~20mA 电流输入与 8 路 0~5V 电压输入。具体电路如图 4.9，4.10 所示。尽管 LPC2134 微控制器内部有 8 路 10 位 A/D 转换器，但作者设计的 PLC 对模拟量输入信号的测量精度要求比较高，因此选用了 CMOS 工艺四位半双积分式 A/D 转

换器 ICL7135。

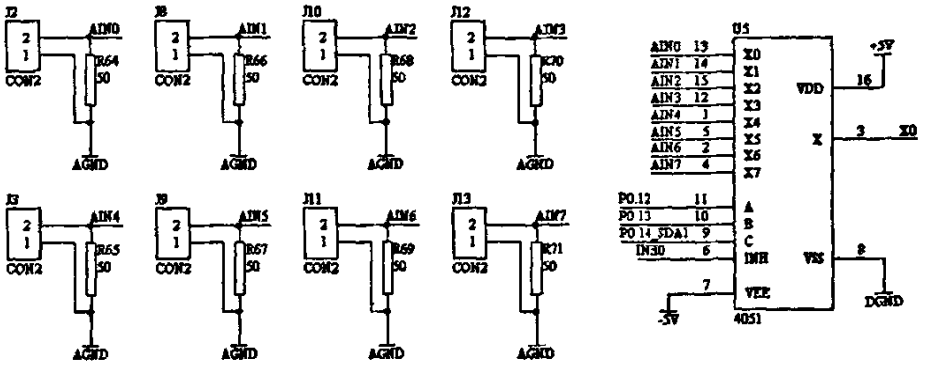

图 4.9　8 路 4~20mA 电流输入

1. 多路模拟开关 CD4051

CD4051 是 8 通道双向模拟开关，内部结构主要由 CMOS 开关、译码电路、电平转换电路三部分组成。模拟开关的外部控制信号（A、B、C 和 INH）由单片机或其他数字电路提供，其电平往往是 TTL 电平，因此须经电平转换电路，将它转换为 CMOS 电平。译码电路根据 A、B、C 和 INH 状态来译码，选通模拟开关中的一个通道。[19]当控制锁存端 INH=1 时，所有通道禁止工作；当 INH=0 时，允许芯片工作，此时 8 个通道的状态将根据 A、B、C 的编码决定。图 4.9 中，CD4051 是 8 入 1 出方式，4~20mA 电流经过 50Ω 电阻转换成 0.2~1V 电压信号，然后通过 CD4051 进行通道选择，从输出端 X 出来的 X0 信号可近似看作 0.2~1V 电压信号。图 4.10 中，CD4051 同样也是 8 入 1 出方式，8 路 0~5V 电压信号分别从 X0~X7 进入 CD4051，由 A、B、C 状态进行通道选择，从 X 端出来的 0~5V 电压信号经由运算放大器 OP07 构成的电压跟随器以及分压电路，从而得到 X1 信号，即为 0~1V 电压信号。X0、X1 信号以及后面的温度检测信号（热电阻、热电偶）一起经过多路模拟开关 CD4051 以及由通用运算放大器 OP07 与多路开关 CD4051 组成的程控增益放大电路（以适应不同输入信号的放大要求），然后经过低通滤波电路，最终送入 A/D 转换器 ICL7135。

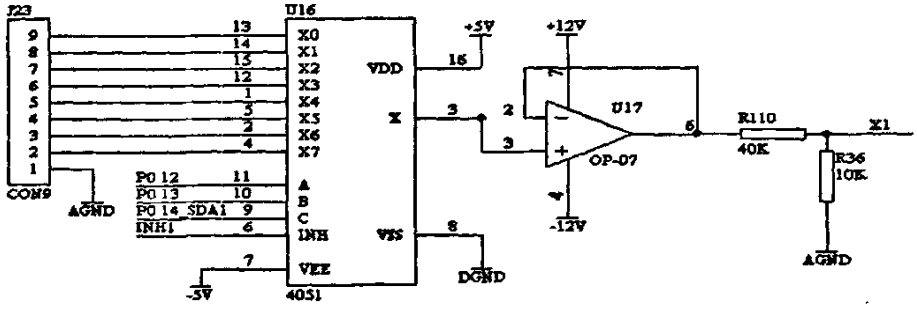

图 4.10　8 路 0~5V 电压输入

2. A/D 转换器 ICL7135

ICL7135 具有 4 位半转换精度，可自动校零点、自动极性输出、动态字位扫描 BCD 码

输出以及自动量程控制信号输出。[21]它的输入电压范围为：-2V~+2V。单极性信号输入时，可达到 14 位以上的分辨率，双极性信号输入时，可达到 15 位以上的分辨率。

ICL7135 A/D 转换的工作时序如图 4.11 所示。整个转换过程可分为四个阶段：即系统调零阶段（10001 个时钟）、信号积分阶段（10000 个时钟）、基准电压积分阶段和积分输出零阶段（共 20001 个时钟），每完成一次转换周期共需 40002 个时钟脉冲，转换结果（实际是上一次的结果）以四位二进制形式的 BCD 码输出（B_8，B_4，B_2，B_1），并同时送出各位的“位同步选通”信号（D_5，D_4，D_3，D_2，D_1）。当 D_5=1 时，B_8，B_4，B_2，B_1 这四个信号对应为万位；当 D_4=1 时，B_8，B_4，B_2，B_1 这四个信号对应为千位；其余依次类推。同时，在每个转换周期，只产生五个负脉冲（由 $\overline{ST}$ 脚输出），其输出时间对应在每个转换周期的五个位选通信号的中间。

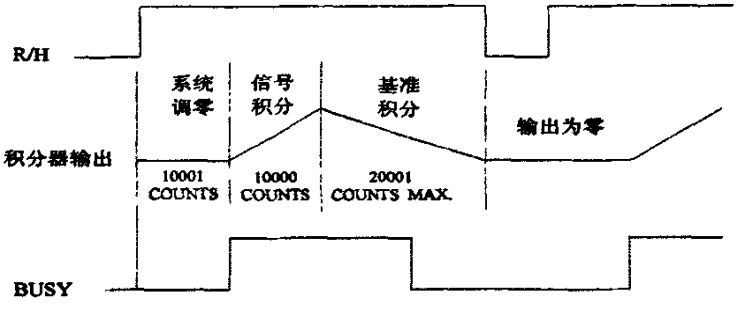

图 4.11 ICL7135 A/D 转换的工作时序

当 R/H 引脚为高电平时，ICL7135 处于连续转换状态；若 R/H 由高电平变为低电平时，ICL7135 进入保持状态。而 ICL7135 是否完成一次转换，可由 BUSY 信号获得。在 A/D 转换期间，BUSY 为高电平，其他时间，BUSY 为低电平。因此，微控制器只要对 R/H 信号和 BUSY 信号进行控制和检测，就能控制 ICL7135 完成 A/D 转换任务。

为了节省系统的硬件资源，特别是微控制器 LPC2134 的 I/O 口，本设计采用了一种比较简洁的硬件连接方式。具体的硬件接口电路如图 4.12 所示。

由 ICL7135 的工作时序可知，用软件在微控制器与 ICL7135 的 R/H 相连的 P0.31 口输出一个正脉冲，则开始启动 A/D 进行转换。转换完后，其不断输出数据。同时把 ICL7135 的 $\overline{ST}$ 引脚与微控制器的 P0.7_EINT2 相连，实现 EINT2 中断。在 A/D 转换期间 $\overline{ST}$ 为高电平；在 A/D 转换结束以后，$\overline{ST}$ 输出五个负脉冲。利用 $\overline{ST}$ 的下降沿请求中断，由于每个 $\overline{ST}$ 负脉冲出现的时刻正是位驱动信号 D_5~ D_1 的中间，同时 B_8，B_4，B_2，B_1 是相应位的 BCD 码，这样，D_5~ D_1 就不必与微控制器相连。软件编程时，连续响应五次 EINT2 中断即为一次转换结果，五次中断均通过与 B_8，B_4，B_2，B_1 相连的 I/O 口读出 BCD 码，依次为转换结果的万，千，百，十和个位。再加上极性信号 POL 引脚，即可得到 A/D 转换结果。

在设计过程中，考虑到 ICL7135 的转换速率与转换精度，ICL7135 芯片外围元器件都严格按照厂家给出的数据，保证参数要求；此外，采用基准源芯片 MC1403 提供参考电压，输出 V_{REF}（ICL7135 选用 1V 参考电压，其满量程为+2.0V），提高 A/D 转换精度。主时钟

频率 f_{CLK} 取 250KHz（晶振频率 f_c 经过分频器 CD4060 得到 f_{CLK}），A/D 转换周期为 40002/250KHz=160ms，即每秒转换 6 次。

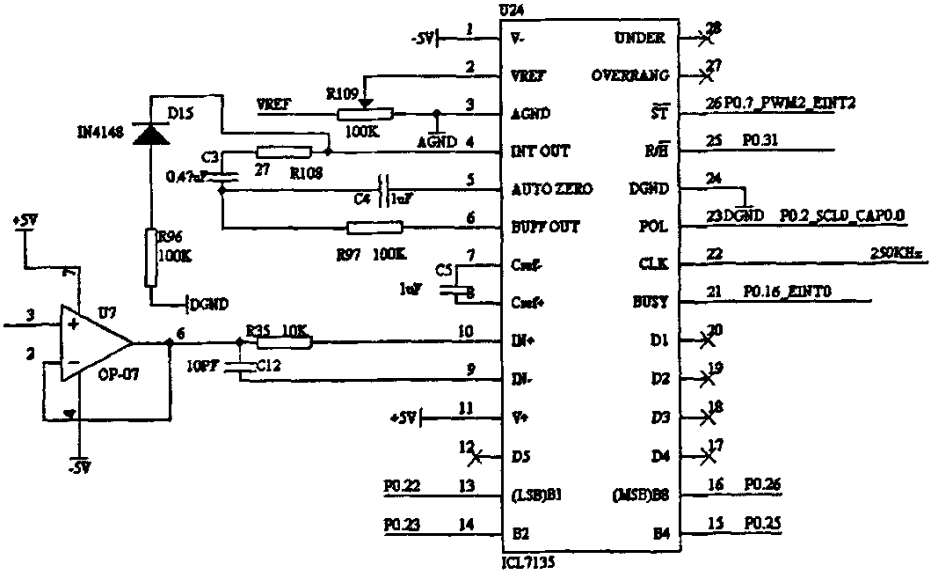

图 4.12　ICL7135 与微控制器 LPC2134 的硬件接口电路

4.5.2 模拟量输出电路设计

模拟量输出电路把 PLC 运算处理的结果（数字量）转换成模拟量，并输出到被选中的某一控制回路上，完成对执行机构的控制动作。通常由控制电路、D/A 转换器、输出保持器、电压/电流转换器、功率放大电路等组成。D/A 转换器是模拟量输出电路的核心电路，它决定了模拟量输出电路的工作速度和转换精度。该 PLC 系统中，模拟量输出电路有 1 路 -10~+10V 电压输出、4 路 0~5V 电压输出以及 2 路 0~20mA 电流输出。下面结合具体电路逐一进行介绍。

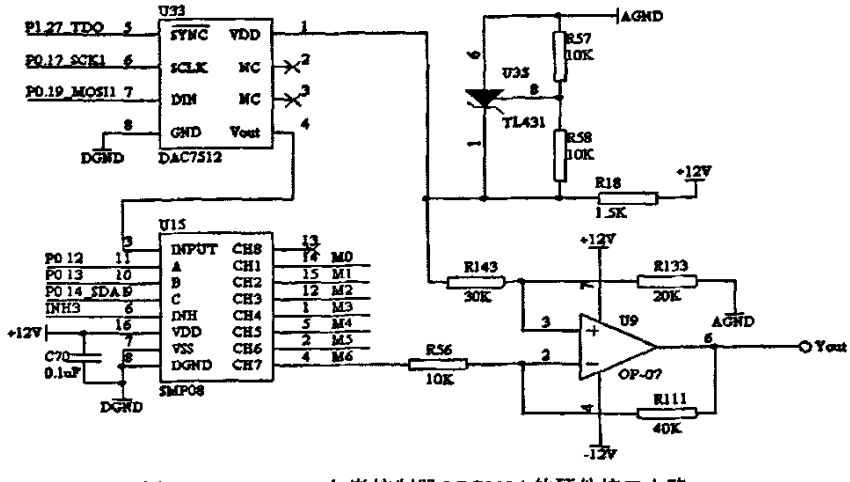

图 4.13　DAC7512 与微控制器 LPC2134 的硬件接口电路

1. 双极性输出电路

DAC7512是串行12位电压输出型D/A转换器，采用与SPI兼容的接口，可以与LPC2134的SPI接口连接。其输出电压 $V_{out} = V_{DD}* (D / 4096)$，其中 D 为输入的二进制代码，它的取值范围为 0~4095。[21]DAC7512 的双极性输出电路如图 4.13 所示。为了提高 D/A 转换精度，DAC7512 参考电压由并联稳压集成电路 TL431 构成的基准电压提供。在 TL431 构成的基准电源电路中，输出电压 $V_0=2.5V*(1+R_{58}/R_{57})=2.5V*2=5V$。因此，DAC7512 的 V_{DD} 即可得到+5V 工作电压。从 DAC7512 的 V_{out} 出来的信号为 0~5V 电压信号。将 V_{out} 出来的信号通过采样保持器 SMP08[22]，可以得到 8 路 0~5V 电压信号。为了提高转换精度，其中 4 路分别经过 4 个电压跟随器便得到 4 路 0~5V 电压信号输出；另外 2 路进入电压/电流转换电路便可以得到 2 路 0~20mA 标准电流信号。此外，还有 1 路即从 CH7 通道出来的信号也为 0~5V 电压信号，将此信号接到由 OP07 构成的差分比例运算电路的反相端，便可实现双极性输出（-10~+10V）。

差分比例运算电路中：假设从输出保持器 SMP08 的 CH7 通道出来的电压为 V_{M6}，

当 $V_{M6}=0V$ 时，此电路实质上就是同相比例运算电路，同相端电压

$$U_P = V_{DD} * (R_{133} / (R_{133} + R_{143})) = 5V * (20/50) = 2V$$

则反相端电压 $U_N = U_P = 2V$。

由 $(Y_{out} - U_N)/R_{111} = U_N / R_{56}$，得到 $Y_{out} = 10V$。

当 $V_{M6}=5V$ 时，该电路就是差分比例运算电路，反相端电压

$$U_N = U_P = 2V$$

由 $(Y_{out} - U_N)/R_{111} = (U_N - V_{M6})/R_{56}$，得到 $Y_{out} = -10V$。

2. 0~5V/0~20mA 转换电路

该设计中，模拟量输出电路有 2 路 0~20mA 标准直流电流信号输出。其原理就是将 2 路 0~5V 电压信号分别通过 2 路电压/电流转换电路，转换成 2 路 0~20mA 直流电流信号。下面就以其中的 1 路为例，对 0~5V/0~20mA 转换电路进行分析。具体电路如图 4.14 所示。

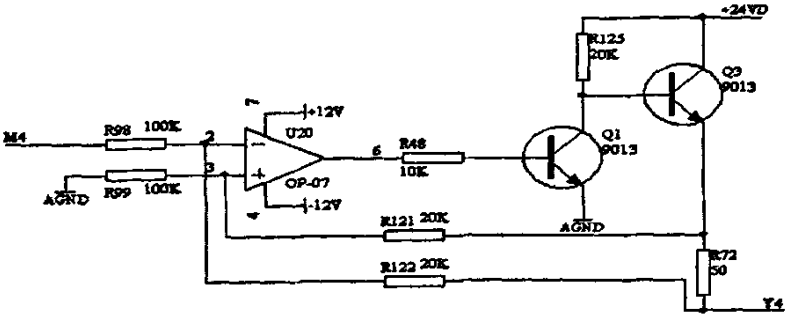

图 4.13　0~5V 信号到 0~20mA 信号的转换电路

假设电阻 R_{121} 与 R_{72} 交点电压为 V_1，M_4 输入电压为 V_i，R_{72} 与 R_{122} 交点电压为 V_2。由

电路可知，运算放大器 OP07 的同相端和反相端电压分别为：

$$V_P = V_1 R_{99} / (R_{99} + R_{121})$$

$$V_N = V_2 + (V_i - V_2) R_{122} / (R_{98} + R_{122})$$

对于运算放大器，$V_P = V_N$，因此有

$$V_1 R_{99} / (R_{99} + R_{121}) = V_2 + (V_i - V_2) R_{122} / (R_{98} + R_{122})$$

由于 $V_2 = V_1 - V_{R72}$，则

$$V_1 R_{98} (R_{98} + R_{122}) + (V_i R_{122} - V_{R72} R_{98}) / (R_{98} + R_{122}) = V_1 R_{99} / (R_{99} + R_{121})$$

其中，$R_{98} = R_{99} = 100K\Omega$，$R_{121} = R_{122} = 20K\Omega$，则有

$$V_{R72} = V_i R_{122} / R_{98} = V_i / 5$$

略去反馈回路的电流，则流经电阻 R_{72} 的电流为：

$$I_o = V_{R72} / R_{72} = V_i / 5R_{72}$$

可见，当运放开环增益足够大时，输出电流 I_o 与输入电压 V_i 的关系只与反馈电阻 R_{72} 有关，因而具有恒流性能。[23] $R_{72} = 50\Omega$ 时，输出电流 I_o 在 0~20mA DC 范围内线性地与 0~5V DC 输入电压对应。从而实现了 0~5V 信号到 0~20mA 信号的转换。

4.6 温度信号采集电路

温度信号采集电路通常由温度传感电路、信号调理电路、A/D 转换电路三部分组成。为了合理使用硬件资源，简化硬件电路设计，该 PLC 系统中温度信号采集电路与模拟量输入电路合用 A/D 转换电路，即使用前面所描述的 ICL7135 来完成温度信号的采集。设计中，温度信号采集电路有热电阻温度传感器采集电路和热电偶温度传感器采集电路两部分组成。下面就结合具体的电路展开介绍。

4.6.1 热电阻温度信号采集电路设计

热电阻是利用导体的电阻随温度变化的特性而制成的测温元件。[24]该部分的温度传感器选用了铂热电阻和铜热电阻。铂热电阻测温范围为-200~+600℃，铜热电阻测温范围为0~+200℃，两者在一定的测温范围都能保证线性的输出特性。两种热电阻测温原理一样，只是不同热电阻与温度的关系不一样。本设计中的铂热电阻采用了 Pt100，铜热电阻采用了 Cu100。其原理就是利用铂（铜）电阻随温度变化阻值发生变化的测温原理，将待测温度量的变化转化为电阻的变化，再进一步通过电桥电路转换成电量的变化。然后经过信号调理电路对信号进行处理，再送入 A/D 转换器 ICL7135 将模拟量转换成数字量，最终送入微控制器 LPC2134。具体测温电路如图 4.14 所示。下面以铂热电阻 Pt100 为例对热电阻测温电路进行简要介绍。

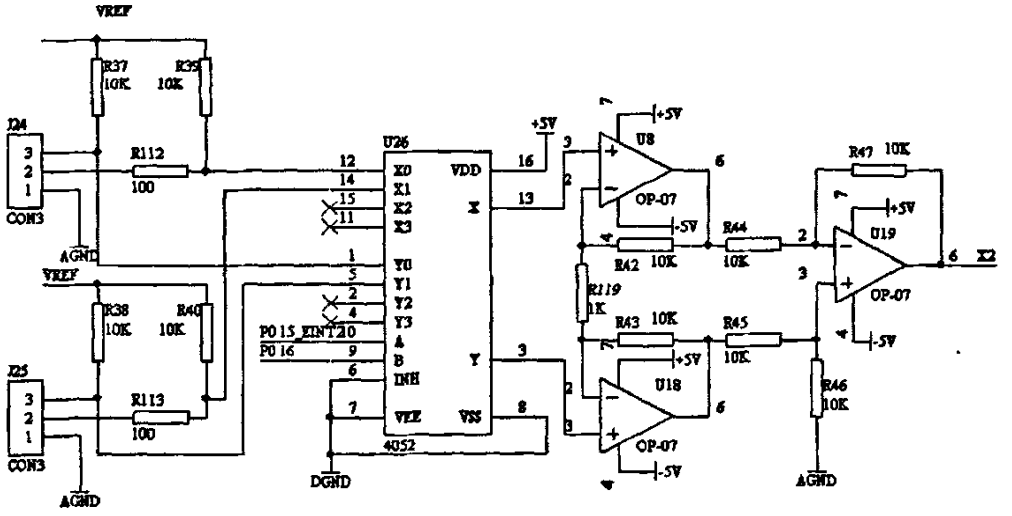

图 4.14 热电阻测温电路

图中 J24 为 Pt100 接入端口，为了保证较高的测量精度和较好的稳定性，采用了三线制铂热电阻测量电路，由电阻 R37、R39、R112 和 Pt100 组成电桥，测温转换电路采用恒压的工作方式，V_{REF} 由 MC1403 型基准电压源提供，这样可以给电桥提供一个合适稳定的工作电压。该电路可以基本消除连线电阻造成的测量误差，适合较重要场合的精密测量。

当温度 t 为-200℃≤t≤0℃时

$$R_t = 100\left[1 + At + Bt^2 + C(t-100)t^3\right]$$

当温度 t 为 0℃≤t≤600℃时

$$R_t = 100\left(1 + At + Bt^2\right)$$

其中，A=3.90802×10^{-3}/℃，B= -5.80195×10^{-7}/℃，C= -4.27350×10^{-12}/℃[25]

本系统中的温度测量电桥的输出电压只有几十毫伏，两输出端均不能接地且存在较大的共模电压，因此后续的放大电路需要采用仪表放大器。为了简便起见，电路中采用由三个运放构成的仪表放大器。[26]

假设运放 U_8 同相输入端输入电压为 V_X，运放 U_{18} 同相输入端输入电压为 V_Y，则 U_8 的输出电压 V_{o1}：

$$V_{o1} = \left(1 + \frac{R_{42}}{R_{119}}\right)V_X - \frac{R_{42}}{R_{119}} \cdot V_Y$$

运放 U_{18} 的输出电压 V_{o2}：

$$V_{o2} = \left(1 + \frac{R_{43}}{R_{119}}\right)V_Y - \frac{R_{43}}{R_{119}} \cdot V_X$$

取 R_{42}= R_{43}，R_{44}= R_{45}，R_{46}= R_{47}，运放 U_{19} 的输出电压 V_o：

$$V_0 = -\frac{R_{47}}{R_{45}}(V_X - V_Y) = -\frac{R_{47}}{R_{45}}\left(1+\frac{2R_{42}}{R_{119}}\right)(V_X - V_Y)$$

设 $V_{Id}=V_X-V_Y$，则

$$V_0 = -\frac{R_{47}}{R_{45}}\left(1+\frac{2R_{42}}{R_{119}}\right)V_{Id}$$

为了保证测量精度，R_{119}选择精密电阻。这样，输出的 X_2 的信号与后续的热电偶温度检测信号一起经过程控放大电路，将信号放大到 A/D 转换器 ICL7135 所要求的输入电压范围，然后送入 ICL7135 进行 A/D 转换，再经过微控制器 LPC2134 到 LED 数码管，就可以显示出温度，同时也可以通过微控制器 LPC2134 对温度进行控制，从而构成一个相对完整的温度测控系统。

4.6.2 热电偶温度信号采集电路设计

热电偶温度传感器相对于热电阻温度传感器而言，具有更宽的温度测量范围（通常从-200~+1600℃）。本设计的热电偶温度信号采集电路可以采集 K 型、J 型、B 型以及 T 型四种热电偶温度传感器的数据。由于不同型号的热电偶传感器输出的信号大小不一，为了合理利用现有资源（即 A/D 转换器 ICL7135），节约硬件成本，使用由 OP07 和 CD4051 组成的程控放大器对传感器信号进行放大处理，放大后的信号进入 ICL7135，经采集处理后得到各路信号值。具体电路如图 4.15 所示。

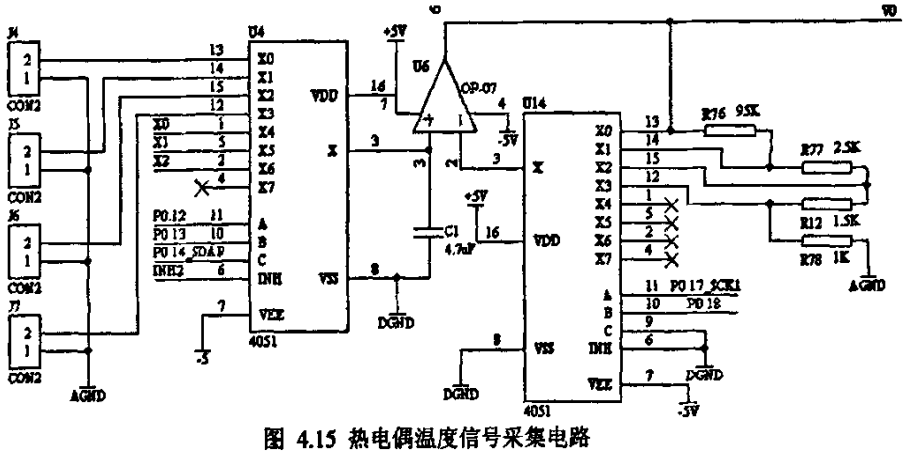

图 4.15 热电偶温度信号采集电路

图中，J4~J7 为四种热电偶的接入端口。通过多路开关 U_4 进入程控放大电路，其中 X_0、X_1、X_2 是前面所提到的电压信号。根据不同的输入信号，选择不同的放大倍数，使的输出电压信号 V_0 满足 ICL7135 的输入范围。程控放大电路利用微控制器 LPC2134 对多路开关 U_5 的控制，选择不同的通道，同时也就选择了不同的放大倍数。

多路开关 U_5 外接的电阻是：R_{76}=95K，R_{77}=2.5K，R_{12}=1.5K，R_{78}=1K。4 个不同的放

大倍数，分别为：

放大倍数 1＝1 倍

$$\text{放大倍数 2}= 1+\frac{R_{76}}{R_{77}+R_{12}+R_{78}}= 1+\frac{95}{5}=20\text{ 倍}$$

$$\text{放大倍数 3}= 1+\frac{R_{76}+R_{77}}{R_{12}+R_{78}}=1+\frac{97.5}{2.5}=40\text{ 倍}$$

$$\text{放大倍数 4}= 1+\frac{R_{76}+R_{77}+R_{12}}{R_{78}}=1+\frac{99}{1}=100\text{ 倍}$$

4 种热电偶作为系统的 4 路毫伏信号输入端。热电偶信号被选通输入后进入程控放大电路，信号分度号不同则毫伏值的高低也不同，通过软件选择不同的放大倍数，使这些放大后的信号最大值接近 ICL7135 的最大允许值。这四个放大倍数完全满足前面四种热电偶输出信号以及本文所涉及的其他电压信号的放大要求。运放 U_6 输出的信号 V_o 经过滤波电路进行滤波，然后就进入 ICL7135 进行 A/D 转换，再经过微控制器 LPC2134 到 LED 数码管显示出温度，也可以通过微控制器 LPC2134 对温度进行控制，从而构成一个相对完整的温度测控系统。

热电偶传感器测温需要进行线性化。通常用硬件实现线性化，即在放大器的输入回路或反馈回路中引入非线性，用线性化器的非线性来补偿传感器的非线性。[27]但使用硬件线性化器，硬件成本加大，线性化精度低，而且不同传感器对应不同的线性化器，若共用同一线性化器必然会增大非线性误差。因此，该设计采用软件线性化，以克服硬件线性化的不足，按传感器的型号设计不同的程序模块并分别调用这些模块即可完成线性化。

4.7 串行通信接口电路

目前几乎所有的微控制器、PC 都提供串行通信接口。[28]本系统为了便于与上位机或其他设备之间的通信，设计了串行通信接口。

串行通信采用的是 RS232 通信标准，其所定义的高低电平信号与 LPC2134 的 UART 接口的 LVTTL 接口电平不同。LVTTL 的标准逻辑“1”对应 2V~3.3V 电平，标准逻辑“0”对应 0V~0.4V 电平，而 RS232 标准采用负逻辑方式，标准逻辑“1”对应-5V~-15V 电平，标准逻辑“0”对应+5V~+15V 电平。[29]显然两者间要进行通信必须经过信号电平的转换。此外由于系统是 3.3V 系统，因此采用 MAXIM 公司的 MAX3232[29]来完成这一功能。

LPC2134 有两个通用异步收发器（UART）接口，它们的结构及寄存器符合 16C550 工业标准。因此，本文选用 UART0 接口通过 MAX3232 与 PC 进行交互。RS-232-C 标准采用的接口是 9 针（DB9）或 25 针（DB25）的 D 型插头，本系统采用的是比较常用的 9 针 D 型插头。要完成最基本的串行通信功能，实际上只需要 RXD，TXD 和 GND 即可。即 9 针串口只连接其中的 3 根线，具体连接电路如图 4.16 所示。其中四个 0.1μF 的去耦电容用来

提高抗干扰能力。

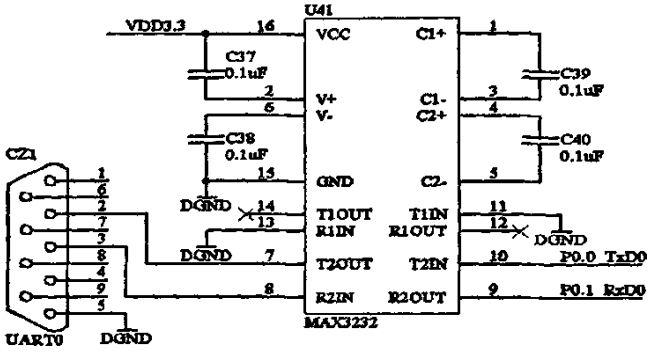

图 4.16　RS232 电平转换电路

4.8 键盘、LED 显示电路

微控制器 LPC2134 提供了硬件 I^2C 总线接口。为了节省 LPC2134 的 I/O 引脚，减轻 LPC2134 用于显示/键盘的工作时间和程序负担，本设计采用键盘与 LED 驱动器 ZLG7290[30] 芯片，它提供了 I^2C 串行接口和键盘中断信号，能方便与微处理器相连。ZLG7290 采用中断方式工作。[31]此外 ZLG7290 能自动完成 8 位 LED 数码管的动态扫描和（最多）64 位按键检测扫描，具有自动消除抖动功能。采用 I^2C 总线方式使得芯片与微控制器的通讯只需 2 个 I/O 口便可完成。需要注意的是：微控制器 LPC2134 的 SDA 和 SCL 端口为开漏输出，所以必须在 SDA 和 SCL 线上分别外接上一个上拉电阻。

4.8.1 键盘电路

本系统采用了矩阵式键盘，设计了 2×8 的键盘，以实现人机交互功能，用户通过键盘可以对被控制参数设定目标值，并可扩展系统功能。具体电路如图 4.17 所示。其中，Dig7~Dig0 为 LED 显示位驱动及键盘扫描线；SegH~SegA 为 LED 显示段驱动及键盘扫描线。

该系统为了能够适应温度数值的处理，设置了 16 个数字键和功能键：

数字键：0~9、小数点按键“.”　功能键：+、—、初始化键、修复键、确认键

ZLG7290 在译码有效的按键后，将按键编号存入键值寄存器中。

按键编码如下：

S2：1，S4：2，S6：3，S8：4，S10：5，S12：6，S14：7，S16：8，S3：9，S5：0，S7：小数点，S9：+，S11：—，S13：初始化，S15：修复，S17：确认。

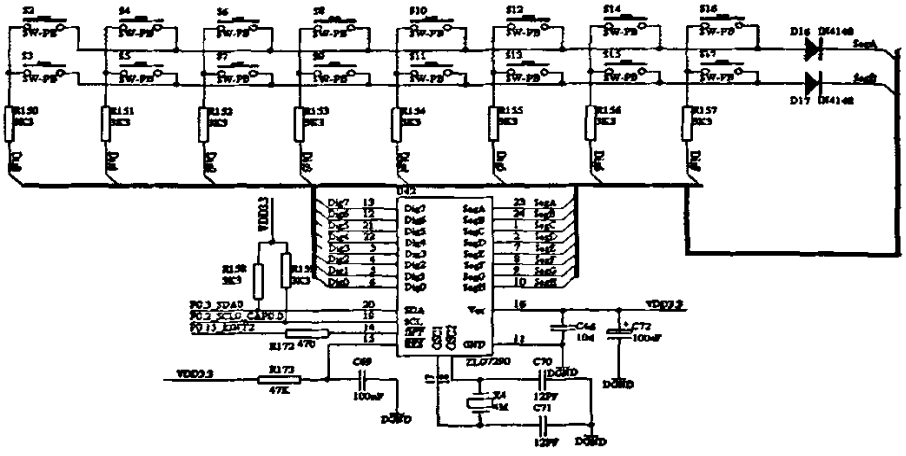

图 4.17　ZLG7290 键盘接口电路设计

4.8.2 LED 显示电路

本系统显示部分用来显示现场温度信号。考虑一般工业现场的工作特点，并结合 ZLG7290 的要求，采用 6 个 8 段共阴极 LED 数码管显示器。数码管在工作时要消耗较大的电流，R161~R168 是限流电阻，为了增大数码管的亮度，取电阻值 220Ω。

4.9 JTAG 接口电路

JTAG（Joint Test Action Group，联合测试行动小组）是一种国际标准测试协议，主要用于芯片内部测试及对系统进行仿真、调试。JTAG 技术是一种嵌入式调试技术，它在芯片内部封装了专门的测试电路 TAP（Test Access Port，测试访问口），通过专用的 JTAG 测试工具对内部接点进行测试。通过 JTAG 接口可对芯片内部的所有部件进行访问，因此是开发调试嵌入式系统的一种简洁高效的手段。

本设计中的 JTAG 接口采用了 20 针接口，其中 TDI、TDO、TMS、TCK，分别为测试数据输入、测试数据输出、测试模式选择和测试时钟。具体电路连接如图 4.19 所示。

根据 LPC2134 的应用手册说明，在 RTCK 引脚接一个 4.7KΩ 的下拉电阻，使系统复位后 LPC2134 内部 JTAG 接口使能，这样就可以直接进行 JTAG 仿真调试。[14]

4.10 蜂鸣器电路

如图 4.20 所示，蜂鸣器使用 PNP 三极管 Q13 进行驱动控制。当 P0.7 控制电平输出 0 时，Q13 导通，蜂鸣器蜂鸣；当 P0.7 控制电平输出 1 时，Q13 截止，蜂鸣器停止蜂鸣。由于 P0.7 口与 SPI 部件的 SSEL0 复用，所以此引脚接一上拉电阻 R63，防止在使用硬件 SPI

总线时由于 SSEL0 引脚悬空而导致 SPI 操作出错。

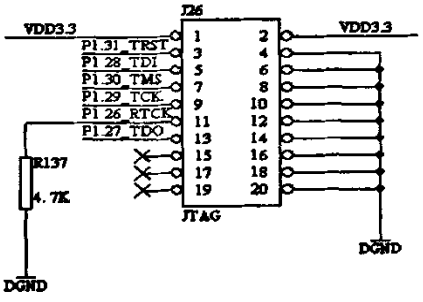

图 4.18　JTAG 接口电路

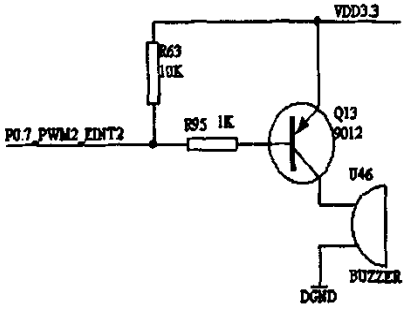

图 4.19 蜂鸣器控制电路

4.11 CAN 总线接口电路

CAN（Controller Area Network，控制器局域网）总线是国际上应用最广泛的现场总线之一。它是一种多主方式的串行通信总线，具有很高的实时性能和很强的纠错能力，支持差分收发，而且传输距离比较远。CAN 属于总线式串行通讯网络，与一般的通讯总线相比，CAN 总线的数据通讯具有突出的可靠性、实时性和灵活性[32]。其具体特征如下：1. CAN 总线的通信方式非常灵活；2. CAN 协议不采用传统的站地址编码方式，而对通信数据块进行编码，用一个 11 位或 29 位二进制数组成的标志符来区分不同的数据块；3. CAN 采用非破坏性总线仲裁技术；4. CAN 的数据帧采用短帧结构，传输时间短；5. CAN 的每帧信息都有 CRC 校验及其他检错措施，保证出错率很低；6. CAN 总线的通信介质多样化，组合方式灵活。目前，CAN 总线已广泛地应用于汽车行业、机器人、智能楼宇、工业控制等领域。

4.11.1 CAN 控制器 SJA1000

CAN 总线的通信协议主要由 CAN 控制器完成。CAN 控制器主要由实现 CAN 总线协议部分与微控制接口部分电路组成。本系统中的微控制器 LPC2134 内部不带片内 CAN 控制器，因此需要设计独立的 CAN 接口电路。经过对几种 CAN 控制器的研究和比较，选择了 PHILIPS 公司的 SJA1000[33]作为本系统的 CAN 控制器。

SJA1000 独立 CAN 控制器，主要用于移动目标和一般工业环境中的区域网络控制。它是 PHILIPS 公司 PCA82C200CAN 控制器(Basic CAN)的替代产品，在完全兼容 PCA82C200 的基础上，增加了一种新的工作模式 Peli CAN，SJA1000 完全支持 CAN2.0B 协议。

微控制器与 SJA1000 之间的状态、控制和命令信号的交换在控制段中完成。微控制器在初始化时将 SJA1000 设置为复位模式，通过对控制段编程以配置通信参数。在运行期间，

微控制器通过读状态寄存器了解网络的状态。[34]在网络通信中所涉及的数据链路层和物理层的操作由 SJA1000 自动完成，无需微控制器干预。因此，在软件设计过程中，只需考虑 SJA1000 的初始化和应用层的设计。

4.11.2 总线接口芯片 82C250

CAN 总线通信接口电路由微控制器、CAN 控制器和 CAN 收发器三部分组成。本文选用了 PHILIPS 公司的 PCA82C250[35]作为 CAN 收发器。

PCA82C250 是 CAN 控制器与物理总线间的接口，从 SJA1000 出来的数据流需经过 PCA82C250 才可与总线相连。PCA82C250 为总线提供差动的发送功能，为 CAN 控制器提供差动的接收功能。使用 PCA82C250 可以增大通信距离，提高系统的抗干扰能力，保护总线，降低射频干扰，实现热保护。

4.11.3 接口电路设计

本文中 CAN 总线接口电路由微控制器 LPC2134、独立 CAN 控制器 SJA1000、CAN 总线收发器 PCA82C250 和高速光电耦合器 6N137 四部分组成。LPC2134 负责 SJA1000 的初始化，通过控制 SJA1000 实现数据的接收和发送等通信任务。具体电路如图 4.21 所示。

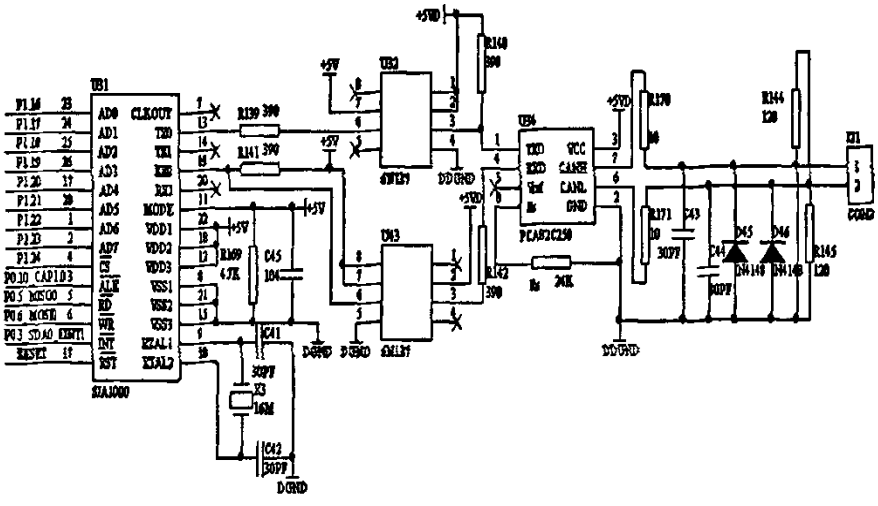

图 4.20　CAN 总线接口电路

SJA1000 是低电平复位，所以直接与微控制器 LPC2134 复位脚相连。SJA1000 的引脚 11（MODE）接高电平，选择 Intel 二分频模式。SJA1000 的引脚 16（INT）与微控制器 LPC2134 的外部中断输入引脚（EINT1）相连。

为了增强 CAN 总线节点的抗干扰能力，SJA1000 的 TXD 和 RXD 通过高速光耦 6N137

后与PCA82C250的TXD和RXD相连，这样能很好地实现总线上各CAN节点的电气隔离。此外，光耦部分采用隔离电源，否则，就失去光耦的作用了。总线两端的两个120Ω的电阻（R113和R114），对于匹配总线阻抗起着相当重要的作用。没有它们，会使数据通信的抗干扰性及可靠性大大降低，甚至无法通信。PCA 82C250的CANH和CANL引脚各自通过一个10Ω的电阻与CAN总线相连，电阻起限流作用，保护PCA 82C250免受过流的冲击。CANH和CANL与地之间并联了两个30pF的小电容，用来滤除总线上的高频干扰，同时可以用来防止电磁辐射。另外在两根CAN总线接入端与地之间反接一个二极管，当CAN总线出现较高的总线电压时，通过二极管可起到一定的过压保护作用。

4.12 硬件抗干扰技术

为了保证系统能够长期稳定地运行，根据系统可能出现的各种干扰，本设计采用了以下几种硬件抗干扰措施：

1. 光电隔离技术

本设计在开关量输入电路中使用光电耦合器TLP521来隔离夹杂在输入开关量中的各种干扰脉冲。在开关量输出电路中，使用光电耦合器来实现系统与被控开关量之间的电气隔离。同样，在CAN总线接口电路中，为了防止总线上的干扰进入系统，在总线控制器SJA1000与总线接口芯片PCA82C250增加了两片高速光电耦合器6N137，这样能很好地实现总线上各CAN节点的电气隔离。需要注意的是：在光电耦合器的输入部分和输出部分必须分别采用独立的电源，否则光电耦合器的隔离作用将失去意义。

2. 串模干扰的抑制

串模干扰指叠加在测量信号上的干扰噪声，一般为变化较快的杂乱无章的交变信号。通常来源于传感器内部或外部引线。在系统模拟量输入电路中，采用双积分式A/D转换器ICL7135来削弱周期性的串模干扰的影响。

3. 电源抗干扰设计

电源滤波和退耦是电源抗干扰的主要措施。电源变压器采用双重屏蔽措施，将初级次级隔离开，减少进入电源的各种干扰。整流滤波电路中采用1000uF电解电容，可以减少高频干扰进入电源系统。整流滤波后得到的直流电压再经过稳压，可使干扰进一步被抑制。

在3.3V电源电路中，用电感L1将模拟电源与数字电源相互隔离（隔离数字电源的高频噪声），以降低噪声与出错几率。

4. 地线干扰的抑制

系统中有多种地线，包括数字地、模拟地、信号地、电源地。为了排除地线的干扰，对这些地线的处理采取“一点接地”原则。不同种类的地线自成体系，然后一点接地。[43]

5. 配置去耦电容

除了电源系统中采用电容退耦电路以外，系统中每块集成电路芯片的电源输入端与接地端间都放置一个0.01uF的瓷片电容。

第五章 嵌入式实时操作系统 μC/OS-II 在 ARM 上的移植

5.1 μC/OS-Ⅱ的文件结构

μC/OS-Ⅱ的文件结构以及与硬件关系如图 5.1 所示。在应用程序中使用 μC/OS-Ⅱ时，需要用户提供应用软件和 μC/OS-Ⅱ的配置部分。核心代码是与处理器类型无关的代码，如任务管理、任务调度、存储管理、信号量、邮箱、消息队列等；[36]配置代码用来配置事件控制块的数目以及是否包含消息管理的相关代码等，应用程序开发人员可以通过修改这些配置文件来裁剪内核，选择自己需要的系统服务；移植代码写的是与处理器类型有关的代码。在移植 μC/OS-Ⅱ操作系统时，主要修改 OS_CPU.H、OS_CPU_A.ASM 和 OS_CPU_C.C 这三个文件。

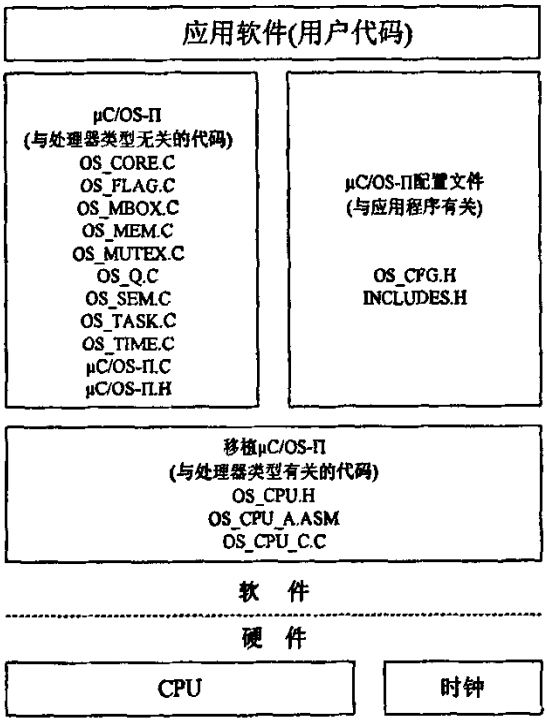

图 5.1　μC/OS-Ⅱ的文件结构

5.2 μC/OS-Ⅱ的移植

5.2.1μC/OS-Ⅱ的移植概述

移植就是使一个实时内核能在某个微处理器或微控制器上运行。为了移植方便，大部

分的 μC/OS-Ⅱ代码用 C 语言编写。移植工作主要集中在 OS_CPU.H、OS_CPU_A.ASM 和 OS_CPU_C.C 三个文件上。其中，OS_CPU.H 文件主要涉及到与编译器相关的数据类型的定义、堆栈类型定义、两个宏定义和几个函数说明；S_CPU_A.ASM 中包含与移植有关的 4 个汇编语言函数，即 OSStartHighRdy(), OSCtxSw(), OSIntCtxSw()和 OSTickISR()；OS_CPU_C.C 包含与移植有关的 10 个 C 函数，除第一个任务堆栈初始化函数外的 9 个函数，需要声明，由系统函数调用，以便用户能在操作系统中加入自己需要的功能。

5.2.2μC/OS-Ⅱ的移植条件

要使 μC/OS-Ⅱ正常运行，处理器必须满足以下要求：

1. 处理器的 C 编译器能产生可重入代码；
2. 用 C 语言就可以打开和关闭中断；
3. 处理器支持中断，并且能产生定时中断（通常在 10—100Hz）；
4. 处理器支持一定数量的数据存储硬件堆栈（可能是几千字节）；
5. 处理器有将堆栈指针和其他 CPU 寄存器内容读出和存储到堆栈或内存中的指令。

5.3 μC/OS-Ⅱ在 LPC2134 上的移植

本设计采用的微控制器 LPC2134 是 ARM7TDMI 内核，具有 ARM 体系的共同特性。支持 ARM ADS1.2 集成开发环境，完全可以生成可重入代码；LPC2134 具有 37 个寄存器，有足够的寄存器资源来关闭或打开系统所有中断；LPC2134 可以产生定时器中断；LPC2134 微控制器有 ARM 和 Thumb 两种指令集，每种指令集都有丰富的指令可以对堆栈进行操作。因此 μC/OS-Ⅱ完全可以移植到 LPC2134 微控制器上。

5.3.1 头文件 INCLUDES.H

INCLUDES.H 是一个头文件，它包含在所有.C 文件的第一行。INCLUDES.H 使得用户项目中每个.C 文件不用分别去考虑它实际上需要哪些头文件，同时增强了代码的可移植性。此外，用户可以通过编辑 INCLUDES.H 来增加自己的头文件，但必须添加在头文件列表的最后。

5.3.2 编写 OS_CPU.H

文件 OS_CPU_.H 中包含用#define 语句定义的与处理器相关的常数、宏以及类型定义。移植时需要在文件中用#define 设置一个常量（OS_STK_GROWTH）的值，声明 10 个数据的

类型，并用#define 声明两个宏（OS_ENTER_CRITICAL( )和 OS_EXIT_CRITICAL( )）。

1. 与编译器相关的数据类型

由于不同的微处理器有不同的字长，所以 μC/OS-Ⅱ的移植包括一系列的数据类型定义，以确保其可移植性。OS_CPU.H 中数据类型定义如下：

```
typedef  unsigned  char  BOOLEAN;
typedef  unsigned  char  INT8U;          无符号 8 位整数
typedef  signed  char  INT8S;            有符号 8 位整数
typedef  unsigned  int  INT16U;          无符号 16 位整数
typedef  signed  char  INT16S;           有符号 16 位整数
typedef  unsigned  long  INT32U;         无符号 32 位整数
typedef  signed  long  INT32S;           有符号 32 位整数
typedef  float         FP32;             单精度浮点数
typedef  double        FP64;             双精度浮点数
typedef  unsigned int  OS_STK;           堆栈入口宽度为 16 位
```

2. OS_ENTER_CRITICAL( )和 OS_EXIT_CRITICAL( )

μC/OS-Ⅱ需要先禁止中断再访问代码的临界段，并且在访问后重新允许中断。这就使得 μC/OS-Ⅱ能够保护临界段代码免受多任务或中断服务例程（ISRs）的破坏。μC/OS-Ⅱ定义了两个宏来禁止和允许中断：OS_ENTER_CRITICAL( )和 OS_EXIT_CRITICAL( )

```
{
    OS_ENTER_CRITICAL( );
    /*  μC/OS-Ⅱ临界代码段 */
    OS_EXIT_CRITICAL( );
}
```

具体实现方法如下：

```
#define  OS_ENTER_CRITICAL( )  disable_int( )    /*  关中断  */
#define  OS_EXIT_CRITICAL( )   enable_int( )     /*  开中断  */
```

此方法就是在 OS_ENTER_CRITICAL( )中调用处理器指令来禁止中断以及在 OS_EXIT_CRITICAL( )中调用允许中断指令。

3. OS_STK_GROWTH

绝大多数微处理器和微控制器的堆栈是从上往下递减的，也有少数处理器使用相反的方式。μC/OS-Ⅱ被设计成两种情况都可以处理，只要在结构常量 OS_STK_GROWTH 中指定堆栈的生长方式即可。

置 OS_STK_GROWTH 为 0 表示堆栈从下往上长；置 OS_STK_GROWTH 为 1 表示堆栈从上往下长。由于 ADS1.2 的 C 语言编译器仅支持从上到下的方式，因此，对于 LPC2134，结构常量 OS_STK_GROWTH 应置 1。如图 5.2 所示。

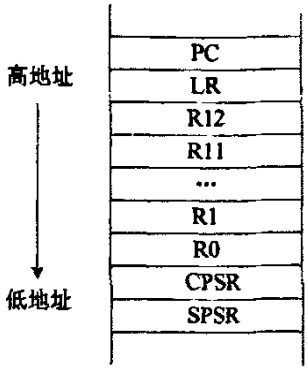

图 5.2 任务堆栈结构示意图

4. OS_TASK_SW( )

OS_TASK_SW( )是一个宏，它是在 μC/OS-II从低优先级任务切换到最高优先级任务时被调用。为了任务调度，通过执行 OS_TASK_SW( )模仿中断的产生，从任务堆栈中恢复处理器所有的寄存器，并且执行中断返回指令。

5.3.3 编写 OS_CPU.A.ASM

这部分代码是对处理器的寄存器进行操作，因此必须用汇编语言来编写。它包括以下4个函数：

1. OSStartHighRdy( )：启动最高优先级任务函数

OSStart( )函数调用 OSStartHighRdy( )来使就绪态任务中优先级最高的任务开始运行。OSStartHighRdy( )假设 OSTCBHighRdy 指向的是优先级最高的任务的任务控制块。μC/OS-II要求处于就绪状态的任务的堆栈结构好象刚发生过中断，并将所有的寄存器保存到堆栈中的情况一样。[14]要运行最高优先级任务，必须将所有处理器寄存器按顺序从任务栈中恢复出来，并且执行中断的返回。这样，就绪的最高优先级任务就如同从中断返回到运行态一样，使得整个系统得以运行。

2. OSCtxSw( ):任务级任务切换函数

OSCtxSw( )在任务级任务切换函数中被调用。任务级切换是通过 SWI 或者 TRAP 人为制造的中断来实现的。[37]它的工作是先将当前任务的 CPU 现场保存到该任务堆栈中，然后获得最高优先级任务的堆栈指针，从该堆栈中恢复此任务的 CPU 现场，使之继续执行。这样就完成了一次任务切换。

OSCtxSw( )的原型

```
Void  OSCtxSw (void)
{
    保存处理器寄存器;
    将当前任务的堆栈指针保存到当前任务的 OS_TCB 中;
```

```
        OSTCBCur->OSTCBStkPtr = Stack pointer;
    调用用户定义的 OSTaskSwHook( );
    OSTCBCur = OSTCBHighRdy;
    OSPrioCur = OSPrioHighRdy;
    得到需要恢复的任务的堆栈指针:
        Stack pointer = OSTCBHighRdy->OSTCBStkPtr;
    将所有处理器寄存器从新任务的堆栈中恢复出来;
    执行中断返回指令;
}
```

3. OSIntCtxSw( ):中断级任务切换函数

OSIntExit( )调用 OSIntCtxSw( )实现中断级任务切换。在时钟中断 ISR（中断服务程序）中，当发现有高优先级任务等待的时钟信号到来时，在中断退出后不返回被中断的任务，而是直接调度就绪的高优先级任务。这样能够尽快让高优先级任务得到响应，保证系统的实时性能。其原理与任务级切换基本上相同，但由于进入中断时已经保存了被中断任务的 CPU
的现场，因此不再进行类似的操作，只需根据函数的嵌套情况对堆栈指针做相应的调整。OSIntCtxSw( )的实现通过设置需要切换的标志位来完成。在 OSIntCtxSw( )里面不发生切换，而是设置一个需要切换的标志，等函数嵌套从 OSIntEnter → OS_ENTER_CRITICAL( ) → OSIntCtxSw( ) → OS_EXIT_CRITICAL( ) → OSIntExit 退出后，根据标志位来判断是否需要进行中断级的任务切换。[38]此方法不需要考虑编译器因素，也无需计算，但实时响应不是最快。

4. OSTickISR( ):时钟中断处理函数

该函数主要负责处理时钟中断，调用系统实现的 OSTimeTick 函数，如果有等待时钟信号的高优先级任务，则需要在中断级别上调度其执行。其他两个相关函数 OSIntEnter( ) 和 OSIntExit( )，都需要在 ISR 中执行。由于 TickISR 也是一种 IRQ 中断，因此只需要在 C_IRQ_hangdler 中调用 OSTimeTick，再开启 Timer 中断就可以了。

5.3.4 编写 OS_CPU_C.C

OS_CPU_C.C 的移植包括任务堆栈初始化和相应函数的实现。要求用户编写 10 个简单的 C 函数。其中，惟一必要的函数是 OSTaskStkInit( )，其他 9 个 Hook 函数又称为钩子函数，主要用来对 μC/OS-Ⅱ进行扩展。

1. OSTaskStkInit( )

OSTaskStkInit( )被 OSTaskCreate( )和 OSTaskCreateExt( )调用来初始化任务的栈结构。在 ARM7TDMI 体系结构下，任务栈空间由高至低依次保存着 PC，LR，R12，R11，…，R1，R0，CPSR，SPSR，如图 5.2 所示。

在用户建立任务时，会传递任务的地址，pdata 指针，任务的堆栈栈顶和任务的优先级给 OSTaskCreate()和 OStaskCreateExt()。为了正确初始化堆栈结构，OSTaskStkInt()

要求刚才得到的前三个参数和一个附加的选项，这个选项只能在OSTaskCreateExt()中得到。如果想从其它函数中调用一个任务，C 编译器就会先将调用该任务的函数的返回地址保存到堆栈中，再将 pdata 参数保存到堆栈或寄存器中。

任务堆栈初始化完成后，OSTaskStkInit 返回新的堆栈指针 stk，OSTaskCreate( ) 执行时将会调用 OSTaskStkInit 的初始化，然后通过 OSTCBInit ( ) 函数调用将返回的 sp 指针保存到该任务的 TCB 块中。初始状态的堆栈其实模拟了一次中断后的堆栈结构，因为任务创建后并不直接就获得执行，而是通过 OSSched ( ) 函数进行调度分配，满足执行条件后才能获得执行。为了使调度简单一致，就预先将该任务的 pc 指针和返回地址 LR 指向函数入口，以便被调度时从堆栈中恢复到刚开始运行时的 CPU 现场。

2. 9个系统 Hook 函数

在 OS_CPU_C.C 文件中，还需要实现 9 个操作系统规定的 Hook 函数：OSTaskCreateHook( )，OSTaskDelHook( )，OSTaskSwHook( )，OSTaskIdleHook( )，TaskStatHook( )，OSTimeTickHook( )，OSInitHookBegin( )，OSInitHookEnd( )，OSTCBInitHook ( )。这些函数都是“钩子”函数，为用户定义函数，由操作系统调用相应的 Hook 函数去执行。一般情况下，必须得声明，但无特殊需求，没有必要包含代码。

第六章 基于ARM的可编程控制器的软件设计

6.1 ADS1.2开发环境简介

ADS集成开发环境（ARM Developer Suite）是ARM公司推出的ARM微控制器集成开发工具，ADS1.2是其成熟版本。ADS1.2支持ARM10之前的所有ARM系列微控制器，支持软件调试和JTAG硬件仿真调试，支持汇编、C、C++源程序；编译效率高，系统库功能较强。ADS1.2集成开发环境由6部分组成，如表6.1所示。

表6.1 ADS1.2集成开发环境的组成

名 称	描 述	使用方式
代码生成工具	ARM汇编器、 ARM的C/C++编译器 Thumb的C/C++编译器、ARM连接器	由CodeWarrior IDE调用
集成开发环境	CodeWarrior IDE	工程管理，编译连接
调试器	AXD、ADW/ADU、armsd	仿真调试
指令模拟器	ARMulator	由AXD调用
ARM开发包	一些低层的例程，实用程序	一些实用程序由CodeWarrior IDE调用
ARM应用库	C、C++函数库等	用户程序使用

本系统软件设计直接操作的就是CodeWarriorIDE集成开发环境和AXD调试器。ADS1.2使用CodeWarrior IDE集成开发环境，并集成有ARM汇编器、ARM的C/C++编译器、Thumb的C/C++编译器和ARM连接器，包含工程管理器、代码生成接口、语法敏感编译器、源文件和浏览器等。AXD调试器为ARM扩展调试器（ARM eXtended Debugger），具有ADW/ADU的所有特性，支持硬件仿真和软件仿真（ARMulator）。AXD装载映像文件到目标内存，具有单步、全速和断点等调试功能，可以便于观察变量、寄存器和内存的数据等。

6.2 LPC2134启动代码的实现

启动代码是芯片复位后进入C语言的main( )函数前执行的一段代码，主要为运行C语言程序提供基本运行环境。启动代码包括异常向量表、堆栈初始化、中断服务程序与C程序的接口、系统初始化代码、异常处理程序和目标板初始化程序等。可以完成堆栈初始化、系统变量初始化、中断系统初始化、地址重映射、I/O初始化以及外围初始化等操作。

编写启动程序是嵌入式系统软件设计的关键，系统启动程序所执行的操作依赖于选择的软件开发平台。本论文在PHILIPS公司提供启动代码基础上，增添了部分与目标板系统

接口相关的代码，使其能够对诸如串口、CAN 以及其它 I/O 端口等进行相应的初始化。为了使得 μC/OS-Ⅱ正常运行，启动代码需要完成如下工作：

1. 定义程序入口

启动程序首先必须定义入口指针，而且整个应用只有一个入口指针。微控制器复位以后，PC 指针指向 0 地址。

2. 设置中断向量表

微控制器 LPC2134 的中断向量位于地址 0x00000000~0x0000001C 处，因此要将中断向量表置于此处。[39]从地址 0x00000000 开始的连续 32 字节的空间分别是复位、未定义指令异常、软件中断、指令预取中止、数据中止、IRQ（外部中断请求）、FIQ（快速中断请求）和一个保留的中断向量。当程序工作于用户 RAM 模式或者外部存储器模式时，需要进行中断向量的重映射。系统在上电和复位时 PC 指针跳到复位异常入口地址处。

3. 中断向量的重映射

调试时，通常程序运行于 RAM 空间（LPC2134 的起始地址为 0x40000000），此时需要进行异常向量的重映射，使得从 0x00000000 开始的 32 字节的异常向量及额外保留的 32 字节映射到 0x40000000 处，同时将存储映射模式配置为用户 RAM 模式。

4. 系统初始化

LPC2134 微控制器的存储系统比较简单，因此系统初始化代码也简单。代码如下：

```
Reset
        BL InitStack ;          /* 调用 InitStack 初始化各种模块的堆栈 */
        BL TargetResetInit;     /* 调用 TargetResetInit 对系统进行基本初始化 */
        B  _main;               /* 跳转到 C 程序入口*/
```

这段代码是程序最终正常运行的流程。首先调用 InitStack 初始化各种模式的堆栈，然后调用 TargetResetInit 对系统进行基本初始化，然后跳转到 ADS 提供的启动代码 _main，它初始化库并最终引导 CPU 进入 main 函数，整个系统就开始正常工作了。[40]

5. 堆栈初始化

编程前首先明确 ARM 使用的堆栈为满降序栈。当系统复位或软件中断响应时，处理器进入管理模式，此时，可以进行各种模块的切换。由此可以设置各种工作模式下的栈指针。初始化堆栈的程序如下：

```
InitStack
    MOV  R0, LR;     /* 当程序退出 InitStack 模块时，处理器模式已为用户模式，因此
LR 不保存返回程序地址，将返回程序地址保存到 R0，同时使用 R0 返回。*/
    MSR  CPSR_c, #0xdb;     /* 设置未定义指令工作模式堆栈 */
    LDR  SP, StackUnd
    MSR  CPSR_c, #0xd7;     /* 设置中止工作模式堆栈 */
    LDR  SP, StackAbt
    MSR  CPSR_c, #0xd1;     /* 设置 FIQ 工作模式堆栈 */
    LDR  SP, StackFiq
    MSR  CPSR_c, #0xd2;     /* 设置 IRQ 工作模式堆栈 */
    LDR  SP, StackIrq
```

```
MSR  CPSR_c, #0xd3;      /* 设置管理工作模式堆栈 */
LDR  SP, StackSvc
MSR  CPSR_c, #0xdf;      /* 设置系统工作模式堆栈 */
LDR  SP, StackUsr
MOV  PC, R0
StackUsr    DCD  (UsrStackSpace+USR_STACK_LEGTH * 4- 4)
StackSvc    DCD  (SvcStackSpace+SVC_STACK_LEGTH * 4- 4)
StackIrq    DCD  (IrqStackSpace+IRQ_STACK_LEGTH * 4- 4)
StackFiq    DCD  (FiqStackSpace+FIQ_STACK_LEGTH * 4- 4)
StackAbt    DCD  (AbtStackSpace+ABT_STACK_LEGTH * 4- 4)
StackUnd    DCD  (UndStackSpace+UND_STACK_LEGTH * 4- 4)
```

6.目标板初始化

由系统初始化代码可知，在进入 main( )函数前，调用函数 TargetResetInit( )对系统进行基本的初始化工作。目标板初始化流程图如下：

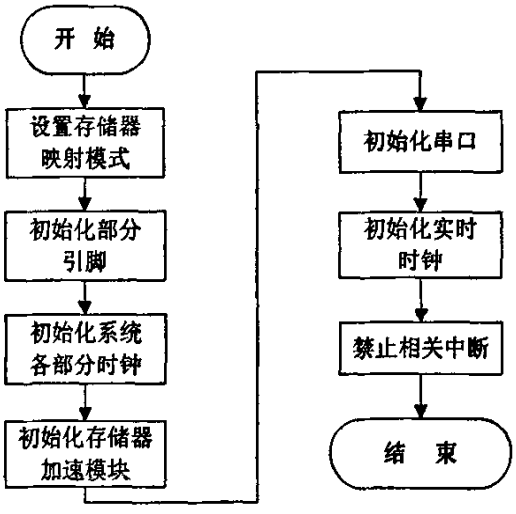

图 6.1 目标板初始化 TargetResetInit( )流程图

一般情况下，系统初始化均在函数 TargeInit( )中完成。但是 μC/OS-Ⅱ要求进入多任务环境之前不允许中断，且进入多任务环境之前使用了一些系统资源，因此需要在 main( )函数之前进行系统的基本初始化。

6.3 系统程序设计

整个系统的软件按功能可分为两部分，即用于管理整个系统正常工作的监控程序和用于执行所要求任务的功能程序（应用程序）。整个系统是在监控程序的控制下进行工作的，因此监控程序的设计是软件设计的核心。应用程序设计采用模块化设计方法。本系统中，应用程序设计主要包括数据采集模块程序设计、A/D 转换中断服务程序设计、D/A 转换模块程序设计、数字量输出模块程序设计、串口通讯模块程序设计以及 CAN 接口通讯模块程序

设计等。本系统着重介绍监控程序设计、A/D 转换中断服务程序设计、D/A 转换模块程序设计、串口通讯模块程序设计以及 CAN 接口通讯模块程序设计。

6.3.1 监控程序设计

监控程序的主要作用就是及时响应来自系统的各种服务请求，有效管理系统自身软、硬件以及人机联系设备，与系统中其他设备交换信息，并在系统出现故障时及时作出相应处理。监控程序通常由监控主程序、初始化管理、键盘管理、显示管理、中断管理、时钟管理、自诊断等模块组成。

其中监控主程序是整个监控程序的一条主线，上电复位后首先进入监控主程序。[41]它管理系统的全部资源。监控主程序的任务是识别命令、解释命令并获得完成该命令的相应模块的入口。监控主程序引导系统进入正常运行，并协调各部分软、硬件有序地工作。

该系统的监控主程序包括系统的初始化、自诊断管理模块、键盘显示管理模块、中断处理程序组成，是“自顶向下”结构化设计的一个层次。其中系统的初始化包括设置中断优先级，各种软件标志的设置以及定时器初始化等。监控主程序的流程如图 6.2 所示。

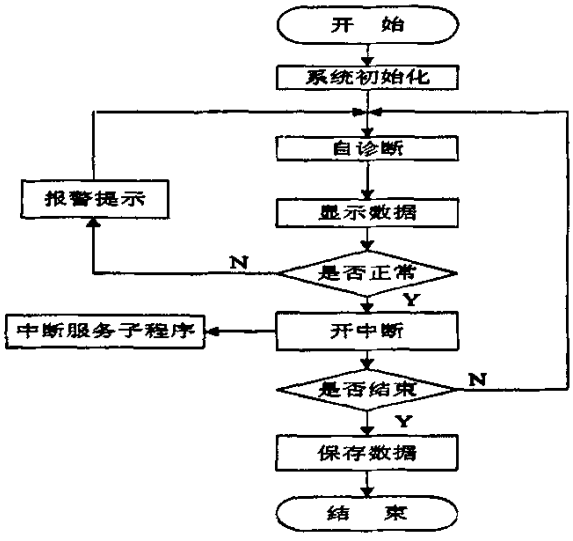

图 6.2 系统监控主程序流程图

6.3.2 A/D 转换中断服务程序设计

根据 ICL7135 A/D 转换原理，将 ICL7135 的 BUSY 端接到微控制器 LPC2134 的外部中断输入引脚，在中断服务程序内完成计数过程。在此过程中，BUSY 端上升沿开始计数，下降沿计数完毕，CPU 通过 ICL7135 的 BUSY 端上的高电平时间得到相应的 A/D 转换的数字值。

LPC2134 通过向量中断控制器(VIC)管理中断。中断的处理流程是：首先，初始化 VIC 使能相关中断，然后正常运行管理程序。当有 IRQ 中断产生，VIC 根据中断源设置

VICvectAddr 寄存器为相应中断服务程序的地址，然后切换处理器工作方式为 IRQ 模式，并跳转到异常向量表的 IRQ 中断入口处，随后读取 VICvectAddr 寄存器的值放入 PC 程序指针，跳转到相应中断服务程序。中断服务程序完成相应中断服务以后，清除中断标志。中断服务结束后，切换回原来的工作模式，并返回原中断点。

本系统 A/D 转换使用外部中断 0 来计数。首先设置 PINSEL1 使 P0.16 连接外部中断源 ICL7135 的 BUSY 引脚，并设置 EXTMODE 为上升沿触发。然后进行中断初始化。设置 VICIntSelect 连接外部 IRQ 中断，设置 VICVectCntl0，连接外部中断到通道 0，把中断服务程序地址放入 VICVectAddr0，即使用向量中断 0，最后在 VICIntEnable 中使能外部中断 0。程序流程图如图 6.3 所示。

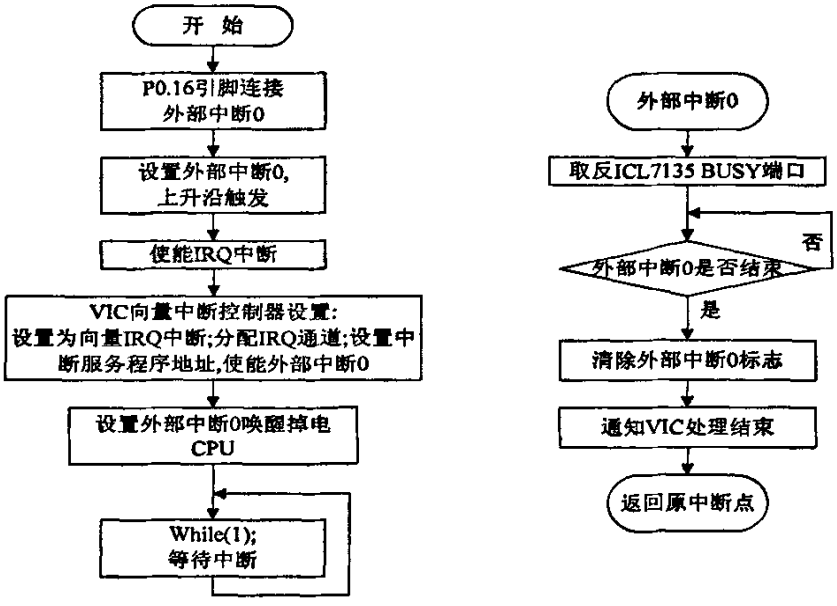

图 6.3 A/D 转换中断服务程序流程图

6.3.3 D/A 转换模块程序设计

由系统硬件设计可知，本系统的 D/A 转换器是具有 SPI 接口的 DAC7512。DAC7512 的 SCLK 引脚、DIN 引脚分别与 LPC2134 的 P0.17、P0.19 连接。P0.17 的 SPP 引脚是 SCK1，它用来同步数据传输的时钟信号，由主机驱动，从机接收；P0.19 的 SPP 引脚是 MOSI1，该信号使串行数据从主机传输到从机。在此程序设计过程中，将 DAC7512 设置为 SPI 从机，将微控制器 LPC2134 的 SPI1 配置为 SPI 主机。其中 DAC7512 的 SPI 接口采用中断的方式接收数据。整个程序设计，首先需要对 SPI 模块的各个寄存器进行初始化，包括使能 SPI，设置时钟分频、相位等；然后进行 VIC 设置，使能 SSP 中断。中断服务程序结束以后，置位接收新数据标志，清除 SSP 中断标志，写 VICVectAddr，最终返回原中断点。程序设计流程图如图 6.4 所示。

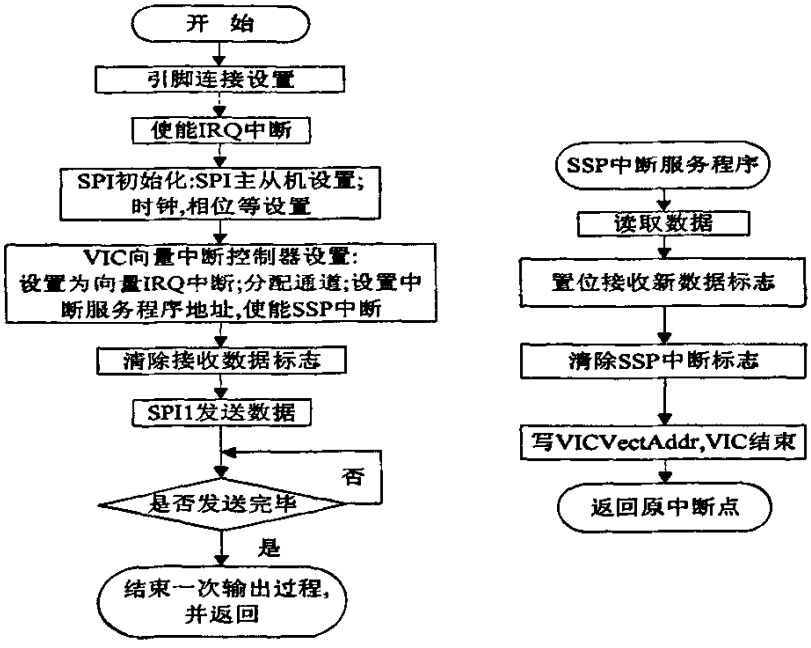

图 6.4 D/A 转换程序设计流程图

6.3.4 串口通讯模块程序设计

LPC2134 有两个符合 16C550 工业标准的异步串行口（UART）UART0 和 UART1。本系统选用 UART0 通过 MAX3232 与 PC 进行交互，采用中断方式进行通信。串口中断服务处理程序主要负责接收、发送数据。中断处理流程图如图 6.5 所示。

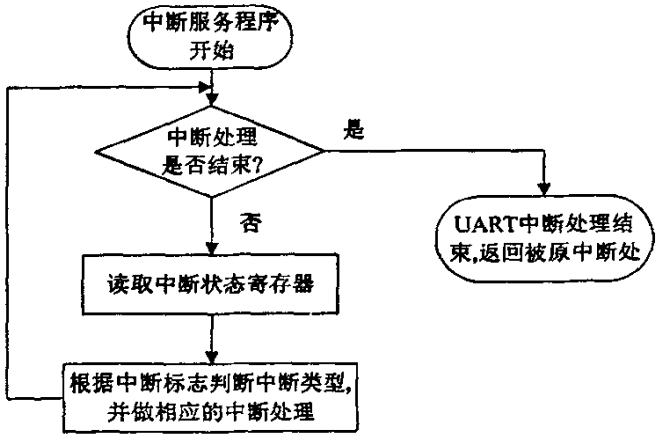

图 6.5 UART0 中断处理流程图

接收数据时应将数据放入 FIFO，并置位接收标志，以便主程序处理；发送数据从 FIFO 按字节取出数据，通过 TxD 引脚发出。

串行口 UART0 初始化程序代码如下：

```
void UART0_init（void）
{
    PINSEL0=0x05;                  // 设置 P0.0、P0.1 分别为 TxD、RxD
    U0CLR=0x08;                    // DLAB=1，可设置波特率
    Bak =  （Fpclk >> 4/ 9600;    // 设置波特率为 9600bps
    U0DLL = bak & 0xFF;
    U0DLM = bak >> 8;
    U0LCR = 0x03;                  // 设置数据传输格式为 8 位数据位、1 位停止位，禁
                                      止
                                   // 奇偶校验，且 DLAB=0
}
```

6.3.5 CAN 接口通讯模块程序设计

CAN 接口通讯程序由三部分组成：CAN 控制器 SJA1000 初始化、CAN 数据发送程序以及 CAN 数据接收程序

1. CAN 控制器 SJA1000 初始化

SJA1000 在系统上电、硬件复位或主控制器发出复位命令后需要进行初始化。初始化包括工作模式的设置，接收滤波方式的设置，接收屏蔽寄存器 AMR 和接收代码寄存器 ACR 的设置，波特率参数设置与中断允许寄存器 IER 的设置等。需要注意的是：在设置有关寄存器前，必须使 SJA1000 由工作模式进入复位模式，否则无法完成初始化操作。此外，本系统中 SJA1000 工作模式采用 Peli CAN 模式。SJA1000 初始化流程图如图 6.6 所示。

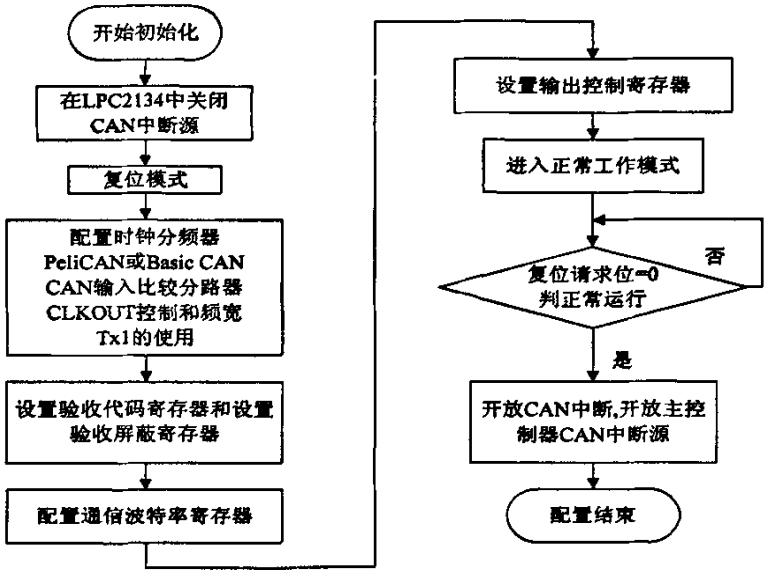

图 6.6 SJA1000 初始化流程图

2. CAN 数据发送程序设计

对 SJA1000 初始化建立 CAN 总线通信后，模块就可以通过 CAN 总线发送和接收 CAN

数据包。消息的发送由SJA1000根据CAN规则自动完成，主控制器必须把要发送的消息送到SJA1000的发送缓冲器中，并设置“发送请求标识位”于命令寄存器中。[42]本设计中CAN总线发送部分的处理采用查询SJA1000控制部分状态标识符的方法。其流程如下图所示。

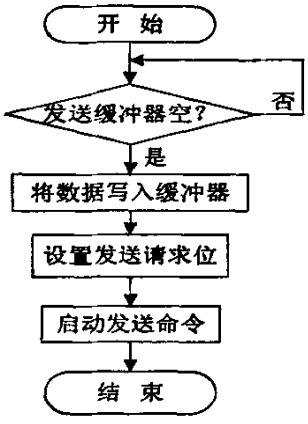

图6.7　CAN数据发送流程图

3. CAN数据接收程序设计

数据接收程序从接收缓冲器读出数据，同时释放接收缓冲器并对数据作出相应处理。本系统中CAN总线数据接收程序采用中断法进行控制。进入中断接收程序，判断中断是不是接收中断；若是，读出接收缓冲器数据并保存，然后清除接收缓冲器。中断控制接收数据流程如图6.8所示。

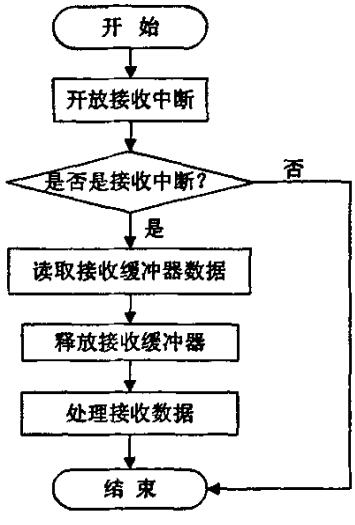

图6.8　中断控制接收数据流程

6.4 PID 控制算法设计

PID 控制是过程控制领域应用最广泛的一种控制规律。常规 PID 控制系统主要由模拟 PID 控制器和被控对象组成。根据给定值 r(t)与实际输出值 c(t)构成的控制偏差

$$e(t)=r(t)-c(t) \tag{6-1}$$

PID 控制规律为

$$u(t)=K_p\left[e(t)+\frac{1}{T_i}\int_0 e(t)dt+\frac{T_d de(t)}{dt}\right] \tag{6-2}$$

传递函数为

$$G(s)=\frac{U(s)}{E(s)}=K_p\left(1+\frac{1}{T_i s}+T_d s\right) \tag{6-3}$$

其中，K_p为比例系数，T_i为积分时间常数，T_i为积分时间常数。

在现代微机测控系统中，PID 控制规律的实现由计算机软件完成，即数字 PID 控制。本设计采用的是增量式 PID 控制算法。在控制算法设计过程中，首先对控制规律进行离散化。r(t)，e(t)，u(t)，c(t)在第 n 次采样时刻的数据分别用 r(n)，e(n)，u(n)，c(n)表示，在采样周期 T 很小时，式（6-2）离散化为以下差分方程式：

$$u(n)=K_p\left\{e(n)+\frac{T}{T_i}\sum_{i=1}^{n}e(i)+\frac{T_d}{T}[e(n)-e(n-1)]\right\}+u_0 \tag{6-4}$$

其中，u_0是偏差为零时的初值。由此可以到 u(n-1)的表达式，即

$$u(n-1)=K_p\left\{e(n-1)+\frac{T}{T_i}\sum_{i=0}^{n-1}e(i)+\frac{T_d}{T}[e(n-1)-e(n-2)]\right\} \tag{6-5}$$

式（6-4）和式（6-5）相减，即得到增量式 PID 控制算法公式：

$$\Delta u(n)=K_p[e(n)-e(n-1)]+K_i e(n)+K_d[e(n)-2e(n-1)+e(n-2)] \tag{6-6}$$

其中，K_p为比例系数，$K_i=K_pT/T_i$ 为积分系数，$K_d=K_pT_d/T$ 为微分系数。

为了编程方便，可将式（6-6）进一步整理为：

$$\Delta u(n)=a_0e(n)+a_1e(n-1)+a_2e(n-2) \tag{6-7}$$

式中，$a_0=K_p\left(1+\frac{T}{T_i}+\frac{T_d}{T}\right)$，$a_1=-K_p\left(1+\frac{2T_d}{T}\right)$，$a_2=K_p\frac{T_d}{T}$

由此可见，增量式算法不需要做累加，控制量增量的确定与最近几次误差采样值有关，计算误差或计算精度问题对控制量计算影响较小。增量式算法得到是控制量增量，误动作影响小。[41]增量式 PID 控制算法的程序流程如图 6.9 所示。

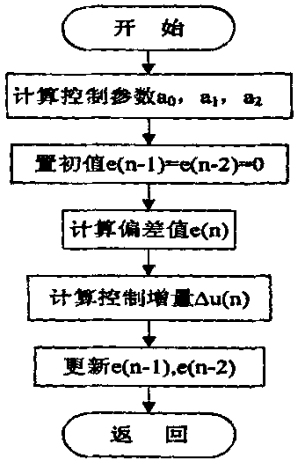

图 6.9 增量式 PID 算法程序流程图

根据控制系统的实际要求，可以对增量式数字 PID 的基本算法进行改进，以改善系统的调节品质。改进仅对基本算法而言，无需增加任何硬件设备。常见的改进算法有不完全微分的 PID 算法，变速积分的 PID 算法，带死区的 PID 算法，微分先行的 PID 算法，纯滞后补偿算法以及采样 PI 控制算法等。由于篇幅有限，就不逐一详细介绍。

6.5 数字滤波程序设计

在微机化测控系统的测量通道中常常存在随机干扰，从而使得经过 A/D 转换后送入微机的数据中存在随机误差。一般情况下可采用硬件抗干扰的方法加以克服。但是对于频率很高或很低的信号，模拟滤波器很难实现。而数字滤波则不存在这样的问题。它可以抑制有效信号中的干扰成分，消除随机误差。数字滤波具有高精度、高可靠性和高稳定性的特点，在智能化测控系统中被广泛地应用于克服随机误差。常见的数字滤波算法有：程序判断滤波、中值滤波、算术平均值滤波、加权平均值滤波、滑动平均滤波、一阶惯性滤波、复合数字滤波等。本系统采集的数据是变化较缓慢的温度值，因此采用是中值滤波。下面就对中值滤波算法进行简单的介绍。

中值滤波就是对某一被测参数连续采样 n 次（一般 n 取奇数），然后把 n 个采样值按大小顺序排列，再取中间值作为本次采样值。中值滤波对于去掉由于偶然因素引起的波动或采样器不稳定所引起的脉动干扰十分有效。[43]进行中值滤波程序设计过程中，首先把 N 个采样值进行排序。排序方法采用“冒泡法”，然后取中间值。中值滤波部分程序如下：

```
# include< stdio.h>
#define  N  5                          /* 设置采样值个数 */
void main ( )
{
   int a[N], i, j, temp;
   for(j=0; j<= N -1; j++)             /*  冒泡法排序  */
```

```
        for(i=0; i<=N - j; i++)
        {
            if(a[i]>a[i+1])
            {
             temp=a[i]; a[i]=a[i+1]; a[i+1]=temp;
             }
         }
        return(a[(N+1)/2]);              /*  返回中值  */
}
```

6.6 线性化处理程序

本系统中的温度信号采集电路涉及到热电阻和热电偶两种温度传感器。它们的输出电压与被测温度之间的特性曲线呈非线性，因此由输出电压求取对应的温度值时，需要进行非线性校正（又称线性化处理）。为了简化硬件电路的设计，同时在保证系统精度的前提下，通常采用软件方法进行校正。采用软件进行线性化的方法主要有查表法、曲线拟合法，后者使用较多的主要有插值法和最小二乘法。[44]

根据本系统中热电阻和热电偶的温度特性，采用线性插值方法可以达到较高的精度，因此本系统的线性化处理采取了线性插值法。

插值法就是从标定或校准实验的 n 组测定数据 (x_i, y_i) (i=1,2, …, n) 中求得函数 g(x) 作为实际输出数据 x 与被测真值 y 的函数关系 y=f (x) 的近似表达式。满足此条件的函数 g(x) 就称为 f(x) 的插值函数，x_i 为插值节点。一般选 g(x) 为 n 次多项式，并记 n 次多项式为 $P_n(x)$,

$$g(x)=P_n(x)=\sum_{i=0}^{n}a_i x^i \qquad (6\text{-}8)$$

线性插值就是在一组数据(x_i, y_i)中选择两个代表性的点 (x_0, y_0) 和 (x_1, y_1)，根据插值原理，可以得到插值方程：

$$P_1(x)=\frac{x-x_1}{x_0-x_1}y_0+\frac{x-x_0}{x_1-x_0}y_1=a_1x+a_0 \qquad (6\text{-}9)$$

其中待定系数 a_1 和 a_0 分别为：

$$a_1=\frac{y_1-y_0}{x_1-x_0} \qquad a_0=y_0-a_1x_0$$

对于非线性程度比较高的情况，采用一个直线方程进行校正很难满足系统精度要求，因此采用分段直线方程进行线性化处理。分段后的每一段非线性曲线用一个方程来校正，即

$$P_{1i}=a_{1i}x+a_{0i} \qquad i=1, 2, \cdots, N \qquad (6\text{-}10)$$

考虑到本系统中的热电偶的温度线性，采用非等距节点分段直线拟合。即在线性较好的部分节点间距离取得大点，反之距离则取得小些，从而使误差达到均匀分布。如图 6.10 所示，用不等分的三段折线进行线性化处理。

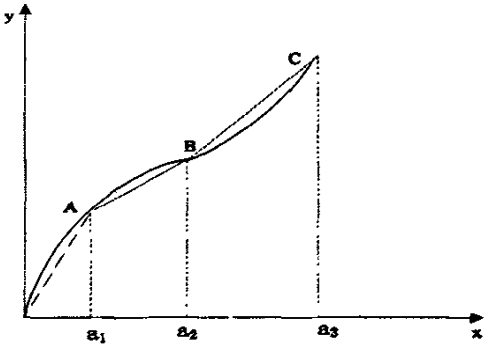

图 6.10　非等距节点分段直线校正

校正方程为：

$$P_1(x)=\begin{cases}a_{11}x+a_{01}\ (0\le x\le a_1)\\ a_{12}x+a_{02}\ (a_1\le x\le a_2)\\ a_{13}x+a_{03}\ (a_2\le x\le a_3)\end{cases}\qquad (6\text{-}11)$$

由此可以求出系数 a_{1i}, a_{0i}（i=1，2，3）。编程时将 a_{1i}, a_{0i} 以及 x_0，x_1，x_3 值一起存于片内存储器中。进行线性化处理时，根据测量值的大小找到所在分段，再从存储器中取出对应段的系数，最后运用方程（6-10）进行计算可以得到实际被测值 y。

6.7 软件抗干扰技术

为了提高系统的可靠性，仅靠硬件抗干扰措施还不够，需要进一步借助软件措施来克服干扰问题。软件抗干扰技术是当系统受干扰后使系统恢复正常运行或输入信号受干扰后去伪求真的一种辅助方法。本设计过程中，软件抗干扰主要涉及的内容是采用软件方法抑制叠加在模拟信号上噪声对数据采集结果的影响。因此采用数字滤波的方法来削弱频率很低的周期性干扰和随机干扰，减少干扰噪声对有用信号的影响。数字滤波采用的是中值滤波法，具体程序设计在 6.5 数字滤波程序设计已经进行过详细描述，此处不再重复。

第七章 结束语

随着微电子技术和计算机技术的高速发展，嵌入式系统技术已经相当成熟。ARM微控制器及其技术已被广泛地应用到工业控制领域以及智能仪器仪表的开发过程中，其高集成度、高性能、低功耗、低成本等突出优点在广泛的实际应用中得到了很好的验证。

通过认真学习研究可编程控制器的基本组成与工作原理，并结合目前国内外可编程控制器的研究状况与发展趋势，本文提出了一种基于ARM微控制器为核心的可编程控制器的研制方案。该方案充分利用ARM微控制器LPC2134的软、硬件资源，结合嵌入式实时操作系统μC/OS-II，借助ARM集成开发工具ADS1.2设计了一种配置简单、扩展方便、可靠性高的小型PLC。系统硬件电路设计方面完成了开关量输入/输出电路设计、模拟量输入/输出电路设计、温度信号采集电路设计、串行通信接口电路设计、键盘显示接口电路设计以及JTAG接口电路设计。此外，为了适应现代PLC网络通信功能，增加了CAN总线接口电路设计。软件设计方面，实现了在ADS编译环境下嵌入式实时操作系统μC/OS-II的移植。在此基础上采用模块化程序设计思想，并借助SmartARM2200教学实验开发平台与μC/OS-II操作系统提供的API函数进行了相关软件的开发，给出了部分模块的程序流程图。为了使得该PLC能够实现更好的闭环控制，在温度信号采集系统中增加了数字增量式PID算法。为了简化系统硬件电路的设计，同时在保证系统精度的前提下，采用线性化处理程序来校正温度信号采集电路中热电阻和热电偶的温度特性。此外，在设计过程中充分考虑了系统的抗干扰问题，从硬件与软件两个方面分别采取了相应的抗干扰措施。

作为一名嵌入式技术的初学者，对嵌入式技术的学习和掌握还不够深入，许多方面只是进行初步的探讨，在以下几个方面需要今后进一步改进和完善。

1. 为了进一步简化系统硬件电路设计，提高系统的可靠性，可以选用自带CAN模块的32位ARM微控制器为核心，比如ARM7TDMI-S系列的PHILIPS的LPC2292/LPC2294。

2. 为了进一步提高数据采集精度，达到更好的控制要求，可以考虑更为先进的控制算法，比如基于BP神经网络的PID控制算法，模糊PID控制算法等。

3. 为了增强PLC的功能，多开发一些特殊功能模块。

鉴于作者水平所限，论文方面难免存在不足之处，敬请各位老师和同学批评指正。

参考文献

[1] 于广庆编著.可编程控制器原理及系统设计[M]. 北京:清华大学出版社,2004.

[2] 陈立定,吴玉香,苏开才等编著.电气控制与可编程控制器[M]. 广州:华南理工大学出版社,2003.

[3] 黄明琪,冯济缨,王福平主编.可编程序控制器[M]. 重庆:重庆大学出版社,2003.

[4] 骆 智.可编程控制器(PLC)运行系统的设计与实现:[硕士学位论文] 北京:北方工业大学检测技术与自动化装置专业,2004

[5] 陈宇,段鑫编.可编程控制器基础及编程技巧[M]. 广州:华南理工大学出版社,2004.

[6] 王仁祥,王小曼编著.现代可编程序控制器网络通信技术[M]. 北京:中国电力出版社,2006.

[7] 骆德汉主编.可编程控制器与现场总线网络控制[M]. 北京:科学出版社,2005.

[8] 杨 柳.基于 PLC 的中药智能配药系统的设计与实现:[硕士学位论文] 四川:四川大学计算机学院计算机应用技术专业,2004.

[9] 马维华主编.嵌入式系统原理及应用[M]. 北京:北京邮电大学出版社,2006.

[10] 徐尼锋.基于32位ARM处理器的嵌入式农业综合测控平台研究:[硕士学位论文] 西安:西安电子科技大学机械电子工程专业,2004.

[11] 于 明,范书瑞,曾祥烨编著.ARM9 嵌入式系统设计与开发教程[M]. 北京:电子工业出版社,2006.

[12] 三恒星科技编著.ARM7 易学通[M]. 北京:人民邮电出版社,2006.

[13] LPC2131_32_34_36_38 Datasheet. Philips Semiconductors. 2005.

[14] 周立功,张华等编著.深入浅出 ARM7—LPC213x/214x（上册）[M].北京:北京航空航天大学出版社,2005.

[15] MC7800 Series Datasheet. MOTOROLA ANALOG IC DEVICE DATA.1997.

[16] MC7900 Series Datasheet. MOTOROLA ANALOG IC DEVICE DATA.1997.

[17] SPX117 Series Datasheet. Philips Semiconductors. 2005.

[18] ULN2003 Datasheet. STMicroelectronics. 2002.

[19] 周振安,范良龙,王秀英,陆小华编著.数据采集系统的设计与实践[M]. 北京:地震出版社,2005.

[20] ICL7135 ANALOG-TO-DIGITAL CONVERTERS Datasheet. Texas Instruments Incorporated. 1999.

[21] DAC7512 DIGITAL-TO-ANALOG CONVERTERS Datasheet. Texas Instruments Incorporated. 2002..

[22] SMP08 Octal Sample-and-Hold with Multiplexed Input. Analog Devices,Inc. 1996.

[23] 李正军编著.计算机测控系统设计与应用[M].北京:机械工业出版社,2004.

[24] 汪 珺.一种铂电阻的高精度温度测量系统[J]. 测试技术. 2004 第 04 期,12~13.
[25] 何希才主编.传感器技术及应用[M]. 北京:北京航空航天大学出版社,2005.
[26] 唐慧强.精密压力变送器的研制[J]. 测控技术. 1999 年 18 卷第 6 期,63~64.
[27] 褚东升.智能型通用数字显示仪的设计[J]. 微计算机信息. 2002 年第 18 卷第 6 期,43~45.
[28] 殷 海.基于 ARM7TDMI 平台下数字信号发生器的设计与实现:[硕士论文] 沈阳:东北大学信息科学与工程学院控制理论与控制工程专业,2006.
[29] Maxim 3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1 μ F External Capacitors Datasheet. Maxim Integrated Products, 1999.
[30] 周立功单片机《ZLG7290I²C 接口键盘及 LED 驱动器数据手册》.http://www.zlgmcu.com.
[31] 唐明军,唐慧强,黄金燕.基于 ARM 的可编程控制器的硬件设计[J]. 仪表技术与传感器. 2006 年第 5 期,40~41.
[32] C. Svelto, G. Galzeran and E. Bava. Compact and accurate digital thermometer based on Anderson's loop and Pt-100 sensor. Measurement Journal of the International Measurement Confederation IMEKO. 2001,NO.2:56~58.
[33] SJA1000 Stand-alone CAN controller Datasheet. Philips Semiconductors，2000.
[34] 徐科军主编.传感器与检测技术[M]. 北京:电子工业出版社,l2004.
[35] PCA82C250 CAN controller interface Datasheet. Philips Semiconductors,2000.
[36] Jean J. Labrosse 著,邵贝贝等译.嵌入式实时操作系统 μ C/OS-Ⅱ（第 2 版）[M].北京:北京航空航天大学出版社,2003.
[37] 严 勇.基于 ARM 和 μ C/OS-Ⅱ的自动气象站实时数据采集系统:[硕士学位论文] 江苏:南京信息工程大学大气物理和大气环境专业,2005
[38] 吴明晖主编.基于 ARM 的嵌入式系统开发与应用[M] 北京:人民邮电出版社,2004.
[39] 孙红波等编著.ARM 与嵌入式技术[M]. 北京:电子工业出版社,2006.
[40] 周立功等编著.ARM 微控制器基础与实战[M]. 北京:北京航空航天大学出版社,2003.
[41] 孙传友,孙晓斌,汉泽西,张 欣编著.测控系统原理与设计[M]. 北京:北京航空航天大学出版社,2002.
[42] 张培仁主编 孙占辉,张村峰,房玉东,张欣编著.基于 C 语言编程 MCS-51 单片机原理与应用[M]. 北京:清华大学出版社,2003.
[43] 薛钧义,武自芳主编.微机控制系统及其应用[M]. 西安:西安交通大学出版社,2003.
[44] 朱自勤主编.传感器与检测技术[M]. 北京:机械工业出版社,2005.